MASTER EN SCIENCES INFORMATIQUES

Mise ONLINE de machines d'analyses biologiques dans le cadre d'un programme de gestion de laboratoire

Wautié, P.

Award date:
1987

Awarding institution:
Universite de Namur

FNDP Namur

Septembre 1987

Mise ONLINE de machines d'analyses biologiques dans le cadre d'un programme de gestion de laboratoire

P. Wautié

Année académique 1986-1987

Promoteur: J. FICHEFET

Je voudrais remercier

- Monsieur FICHEFET, promoteur de ce mémoire
- Monsieur LECLERC, pour ces critiques constructives
- Messieurs DUBOIS et JANSSENS, directeurs du laboratoire de biologie médicale de Nivelles, pour leur accueil, leurs explications et la confiance qu'ils m'ont témoignée lors du stage
- Messieurs ROULIN et BOUCHAT pour leur aide en informatique
- tous les employés du laboratoire.

Je remercie tout spécialement Messieurs DIEU et PIERSON pour leurs précieux conseils lors de la connection hardware.

J'apprécie à sa juste valeur le concours de P. Quertinmont, P. Monsanto, M. Leclerc, et bien d'autres...

ABSTRACT

ABSTRACT

Dans ce mémoire nous checherons à améliorer le fonctionnement d'un laboratoire d'analyses médicales: Laboratoire JANSSENS et DUBOIS de Nivelles. Nous réaliserons la connection (mise ONLINE) de machines d'analyses au programme de gestion du laboratoire préexistant.

Nous commencerons par donner quelques explications indispensables à la compréhension de la suite du mémoire.

- L'interface RS232C est utilisée ici entre deux DTE sans utilisation de DCE. L'étude technique, procédurale, élctrique et fonctionnelle est indispensable dans la connection de nouvelles machines.
- Les machines d'analyses sont de véritables automates travaillant suivant 3 logiques précises. Chaque machine permet de transmettre ses résultats par interface RS232c suivant ses normes et formats propres.
- Le système UNINA permet le transfert des résultats via un concentrateur. Les protocoles utilisés entre le host et le concentrateur sont du type BSC ou HDLC suivant le caractère unidirectionnel ou bidirectionnel de la communication.

Dans notre cas, l'analyse fonctionnelle était inutile puisque la décision de mettre les machines ONLINE avait été prise par le laboratoire et que les choix technologique étaient décidés (système UNINA). Cette analyse permet toutefois de réaliser une comparaison entre cinq solutions existantes et aussi de mettre en évidence quelques causes d'insatisfaction non évidentes. La modélisation de tout ceci a facilté la programmation.

L'étude informatique (ou méthodologie de développement) va nous prouver la necessité de la réalisation de deux programmes distincts:

1. Un programme assurant la réception des données en provenance du concentrateur UNINA.
2. Un programme donnant la possibilité de vérifier et corriger les résultats après leur transfert.

Ces deux programmes ont été réalisés en suivant la méthode préconisée par Alex Van Lamsweerde. Les tests de validation et d'intégration se sont déroulés avec succès.

La réalisation informatique (implémentation) n'entre pas dans le cadre de ce mémoire et n'y sera donc pas développée (En annexes, vous trouverez les textes des programmes, des données et modes d'emploi).
Toutefois, il est intéressant de reprendre le problème après l'implémentation et ainsi de mettre en lumière certaines critiques telles que

- utilisation non-optimale des pointeurs
- choix des fichiers qui se sont révélés lourds à gérer
- Utilisation intéressante de système de bas niveau
- Améliorations possibles
- Utilisation massive de la mémoire pour la rapidité
- Etc ...

La mise en place des programme s'est révélée difficile et a provoqué quelques réflexions et affirmations qui font suite à une discussion avec le Père Berleur

- Installation : nécessité de diplomatie
- Présentation : simplicité et efficacité
- Développement : collaboration avec l'utilisateur

La conclusion sera conduite en trois points:

- Les objectifs du laboratoire sont satisfaits
- Ce mémoire permet la maintenance du programme et l'ajout de nouvelles machines
- Dans le futur, pourquoi ne pas penser à reprendre le système le plus simple: la connection directe.

INTRODUCTION

INTRODUCTION

"Mise ONLINE de machines d'analyses biologiques dans le cadre d'un programme de gestion de laboratoire" n'est pas une thèse; innovations et découvertes existentielles n'y foisonnent certes pas.

Ce mémoire, c'est ce qui nous restera de notre première confrontation avec la vie d'entreprise.

Le contexte : entreprise en pleine phase d'informatisation. Il va sans dire que l'application dont on nous a confié le soin devra parfaitement s'intégrer dans ce renouveau informatique.

L'entreprise : Laboratoire de biologie médical. -Nivelles-

L'application : cfr titre.

l'échéance : Janvier 87.

Les précisions : étant données des machines d'analyses biologiques dotées d'une interface rendant possible le transfert automatique de leurs résultats dans divers programmes de gestion du laboratoire, on demande d'écrire un programme de connection et un programme capable de vérifier et de corriger si nécessaire les résultats transférés. Les erreurs peuvent non seulement provenir d'un mauvais transfert mais aussi d'une mauvaise manipulation du personnel du laboratoire.

Le matériel conseillé : le système UNINA : système de transfert et de concentrateur spécifiquement destiné aux machines d'analyses médicales.

Première approche du problème : familiarisation avec la structure dans laquelle notre application devra s'inscrire, avec le matériel qu'elle devra exploiter et finalement avec les concepts sur lesquels elle devra s'appuyer.

Réalisation de l'application : L'approche globale du problème ne peut être mieux illustrée que par un survol des chapitres qui suivent, il montre quelle fut la logique suivie et les outils dont on s'est servi.

Survol des différents chapitres

Le premier chapitre -Quelques explications préalables- est à notre sens un prérequis à une complète compréhension du problème. On y trouvera plus particulièrement un synoptique de la structure et de l'organisation de l'environnement, une définition du ONLINE, certains aspects techniques des machines d'analyses ainsi qu'un exposé de la norme RS232c et du système UNINA.

Nantis de ces divers éléments, nous pourrons dès lors, nous lancer dans une analyse fonctionnelle de l'application. Précisons simplement ici que nous avons choisi de suivre la méthode conseillée par Messieurs Bodart et Pigneur dans leur livre "Conception assistée des applications informatiques" -Presse Universitaire de Namur.

L'analyse fonctionnelle nous conduisant à un éventail de solutions, nous en choisirons une et lui appliquerons la 'Méthodologie de programmation' préconisée par Monsieur Van Lamsweerde -professeur aux FNDP-.

C'est de ces différentes étapes que découlera l'implémentation proprement dite.

Plutôt que de verser dans un exposé ennuyeux des moindres détails -astucieux ou non- de la mise au point de notre programme nous donnerons les principaux axes autour desquels il s'articule tout en relevant les grands choix informatiques auxquels nous avons dès lors été confrontés

Vient alors une critique aussi objective que possible de notre travail. Plus précisement, nous remettrons en question notre structure de données et nos choix à priori, et relèverons quelles pourraient être les améliorations souhaitables à apporter à notre solution.

Dans le dernier chapitre enfin nous évoquerons les difficultés rencontrées lors de la mise en place de l'application et nous tâcherons d'en retirer quelques enseignements.

La conclusion quant à elle, n'est autre qu'un bilan de l'opération 'mise ONLINE de machines d'analyses'. Notre verdict:positif.

Au début des chapitres qui s'y prêtent, se trouve un résumé des explications données dans la suite. Ainsi vous pourrez ne pas vous attarder sur des développements que vous pensez connaître.

Remarque:

La publicité est interdite dans le cadre de la promotion d'un laboratoire biologique. Certains points devront, dès lors, être considérés comme éléments nécessaires à la compréhension du mémoire plutôt que comme publicité illicite.

CHAPITRE I

QUELQUES EXPLICATIONS PREALABLES

LE LABORATOIRE

En quelques mots

Le laboratoire de Nivelles est une société de services. Son objet est la prestation d'actes techniques relevant de la biologie clinique. Un soin particulier est apporté tant à la qualité des analyses qu'à la rapidité avec laquelle les résultats sont transmis aux médecins. Ces soucis constants sont des aprioris nécessaire à une gestion dynamique du laboratoire. Dans cette optique, des modifications structurelles importantes ont été apportées afin d'aboutir à une amélioration substantielle des services.

En détail :

Il n'est pas difficile de prouver que ce laboratoire est une entreprise (unité de production) comme telle. Elle se décompose en services que nous étudierons plus tard et son but est de réaliser et vendre les résultats d'analyses médicales. Mais au fait, savez vous ce qu'est une analyse?
Une analyse est l'ensemble des modes opératoires conduisant à l'identification qualitative et/ou à la mesure quantitative de substances chimiques ou de structures cellulaires. Les qualités d'une méthode d'analyses dépend des critères suivants

- EXACTITUDE : écart entre la valeur mesurée et la vraie valeur.
- PRECISION : dispersion des mesures effectuées sur un même échantillon
- FIABILITE : capacité de maintenir dans le temps exactitude et précision.

Plus précisement encore, une analyse médicale est l'ensemble des recherches morphologiques, physiques, chimiques et biologiques qu'il est possible d'effectuer sur des tissus organiques, ou des parcelles de tissus, sur des sécrétions, excrétions ou produits pathologiques de l'organisme pour y déceler la présence d'éléments pathologiques éventuels et, en déterminer les qualités et quantités.

Le laboratoire est décomposé en services. (centres d'activités plus ou moins autonomes)

- la <u>direction</u>

La sphère de direction est composée de trois personnes se partageant les responsabilités et décisions importantes.

- Le <u>secrétariat</u>

Ce service partage son temps entre de nombreuses tâches qui semblent bien hétérogènes, à savoir

* accueillir les patients et les médecins porteurs d'une demande d'analyses
* téléphoner les résultats urgents aux médecins

* recevoir les appels téléphoniques émanant de patients ou de médecins
* étiqueter les échantillons et les demandes d'analyses
* enregistrer / modifier les fiches signalétiques des patients
* entrer les demandes d'analyses
* encoder les résultats d'analyses
* éditer les réponses qui seront envoyées aux médecins.
* éditer les factures et attestations de soins.

- Le laboratoire comme tel

Ce service, de loin le plus important en nombre, se compose de douze personnes réparties en cinq sous-services : chimie, hématologie, R.I.A., sériologie, virologie. Ces cinq services proposent une palette de 247 analyses différentes. Ceci n'est parfois pas suffisant et il convient alors de faire appel à des laboratoires extérieurs. Les techniciens doivent

* constituer les listes de travail
* effectuer les analyses
* inscrire les résultats des analyses
* effectuer les commandes de produits
* réceptionner les arrivages de produits
* verifier la facture

- Le service comptable

Ce service compte deux personnes qui se consacrent exclusivement aux problèmes administratifs et comptables. Ces deux personnes procèdent à

* l'enregistrement des opérations comptables
* l'édition des états comptables
* la récolte des informations nécessaires auprès du patient
* la récupération des paiements de factures en souffrance

L'organisation de tout le laboratoire se base sur l'identification des échantillons et des demandes d'analyses. Cette identification était constituée d'une lettre (a,b,...l) pour le mois et d'un nombre attribué dans l'ordre chronologique d'arrivée des échantillons. Dans la nouvelle informatisation, cette identification a changé et est uniquement constituée d'un nombre de 6 chiffres toujours attribué dans l'ordre d'arrivée. Ces identifications seront le seul moyen de retrouver à quels échantillons correspond une demande d'analyses et vice-versa.

Pour parfaire la compréhension du mode de fonctionnement du laboratoire, je vous propose de suivre une demande d'analyses et son échantillon au travers des étapes du laboratoire.

L'échantillon et sa demande d'analyses sont acheminés vers le secrétariat (Ce sont les patients eux-même ou les médecins qui les apportent). Dans ce service, les deux composants vont être marqués d'un numéro d'identification puis ils seront transmis au service laboratoire. Une personne de ce service va alors "programmer" les analyses demandées sur des joblistes (c-a-d qu'on note sur feuilles les analyses demandées pour un échantillon. Chaque feuille (ou jobliste -voir annexe-) est destinée à un et un seul sous-service. Ces joblistes sont organisées sous forme de tableaux ayant comme identification de ligne le numéro d'identification attribué par le secrétariat et comme identification de colonne le code de l'analyse. A l'intersection des deux on trace une ligne si l'analyse est demandée et c'est à cet endroit que le résultat sera noté. Quand toutes les demandes d'analyses ont ainsi été "programmées" on fait parvenir les joblistes et les échantillons dans les services concernés. Les analyses sont alors effectuées (la plupart se font dans les 24 heures) et les résultats notés sur ces joblistes. Pendant ce temps, les demandes d'analyses sont retournées au secrétariat qui les encode

(les analyses demandées ainsi que des renseignements mutuelle nécessaires) Quand les résultats sont retournés au secrétariat, ils sont encodés. Après cet encodage, les protocoles (résultats et commentaires) peuvent être sortis sur imprimante. Il restera alors aux directeurs à les signer et à les transmettre aux médecins demandeurs. Les échantillons sont congelés et conservés une semaine encore alors que les demandes seront archivées pendant plusieurs années. Il sera encore nécessaire de comptabiliser les analyses et d'envoyer le décompte final aux différentes mutuelles qui remboursera les analyses. (régime du tiers payant)

Ce parcours est un modèle du genre. De nombreuses anomalies de parcours peuvent apparaître:

- analyse oubliée
- résultats invraisemblables
- analyse irréalisable
- renseignements mutuelle incomplets
- etc...

Tous ces problèmes font partie du quotidien et il sera important d'en tenir compte plus tard.

ONLINE

Le mot ONLINE n'est en fait que la traduction anglaise du mot "connection".
Dans notre cas, nous essayerons de "connecter" des machines d'analyses et un système informatique. Cette connection suppose une communication entre ces deux équipements. Or une communication n'est possible que s'il existe une compatibilité hardware et software de ces équipements. Il est à noter qu'il existe un très (trop) grand nombre de systèmes de communication hardware et software. Actuellement pourtant quelques organismes internationnaux (entre autre le Comite Consultatif International Téléphonique et Télégraphique -CCITT- ou le Electrical Industry Association -EIA-) tentent de définir une norme pour faciliter la mise en communication entre équipements. Lorsque cette norme sera acceptée et appliquée par les constructeurs de machines d'analyses ainsi que par les constructeurs d'équipement informatique, il sera alors presque trivial de réaliser ces connections. La communication que nous devons effectuer suppose des échanges d'informations. Ces échanges peuvent être de deux types :

* UNI-directionnel, c'est à dire qu'un seul des équipements connectés transmet des informations pendant que l'autre les reçoit (il est à remarquer que si le récepteur ne recoit pas correctement le message pour une raison quelconque, il lui est impossible de le faire savoir à l'émetteur puisque la communication dans ce sens n'existe pas)
* BI-directionnel, dans ce cas, les équipements peuvent communiquer entre-eux dans les limites de ce qui est prévu par l'algorithme de communication de ces équipements.

LES MACHINES D´ANALYSES

En quelques mots

Quels que soient leurs systèmes de fonctionnement (spectrométrie, néphélométrie,..) les machines d´analyses donnent leurs résultats sous forme numérique. Ces résultats sont alors interprétés. Certains automates d´analyses médicales peuvent transmettre les résultats par interface RS232 selon un format précis et différent pour chaque machines. Quelques-uns de ces automates peuvent même être programmés via cet interface

Pour plus de précisions

Technique

En utilisant des sytèmes complexes tels que photométrie, néphélométrie, comptage d'éléments figurés , électrophorèse, etc ... il est possible de réaliser des analyses en limitant les manipulations humaines. Les résultats sont donnés sous format numérique, les laborantins peuvent ainsi interpreter ces résultats et recommencer certaines analyses qui semblent improbables.

Automate

La plupart de ces machines d'analyses utilisent un algorithme pour réaliser les analyses. Ces algorithmes sont modifiables et il est donc possible de programmer un grand nombre d'analyses sur une seule machine. Vu cette possibilité de programmation, j'emploierai indifféremment les termes machines d'analyses ou automates d'analyses.

Fiabilité

En début de journée, toutes les analyses sont effectuées sur un échantillon dont les résultats sont connus. En comparant les résultats obtenus aux résultats attendus, on peut décider de remplacer certains réactifs, nettoyer la machine ou faire venir le service d'entretien. Ceci permet une grande fiabilité des analyses.

Programmation

Les machines permettent un grand nombre d'analyses mais il n'est pas nécessaire de toutes les réaliser sur tous les échantillons. Il faut donc définir l'ensemble des analyses à effectuer pour un échantillon particulier. C'est ce qu'on appelle "programmer" la machine d'analyses. Cette programmation doit alors se faire par un clavier intégré avant que les analyses ne débutent. Lorsque cette programmation est terminée et les échantillons placés, l'automate travaille seul (ceci dans la plupart des cas).

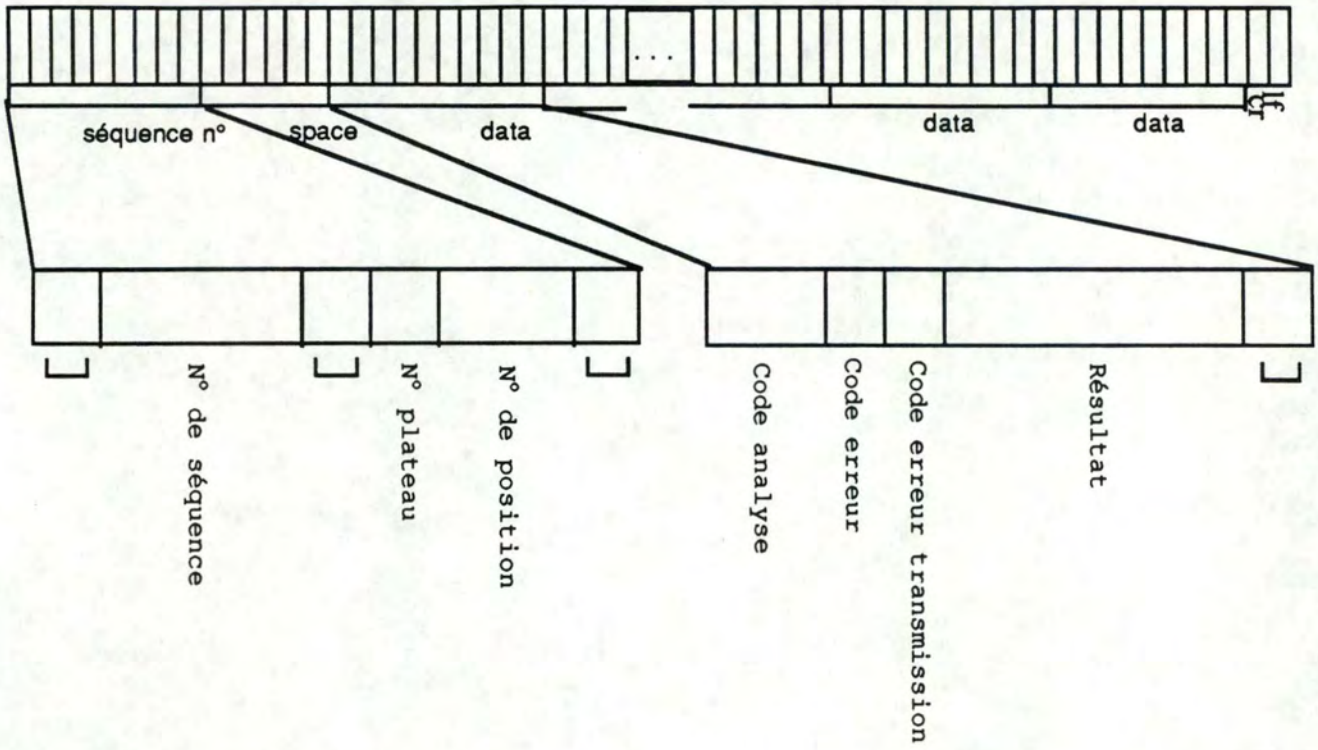
Les résultats

Les résultats des analyses sont communiqués par un système intégré : une imprimante et/ou un vidéo. Le laborantin devra vérifier les résultats (voir s'il ne sortent pas des normes de vraisemblance auquel cas les analyses sont recommencées) et les recopier sur papier.

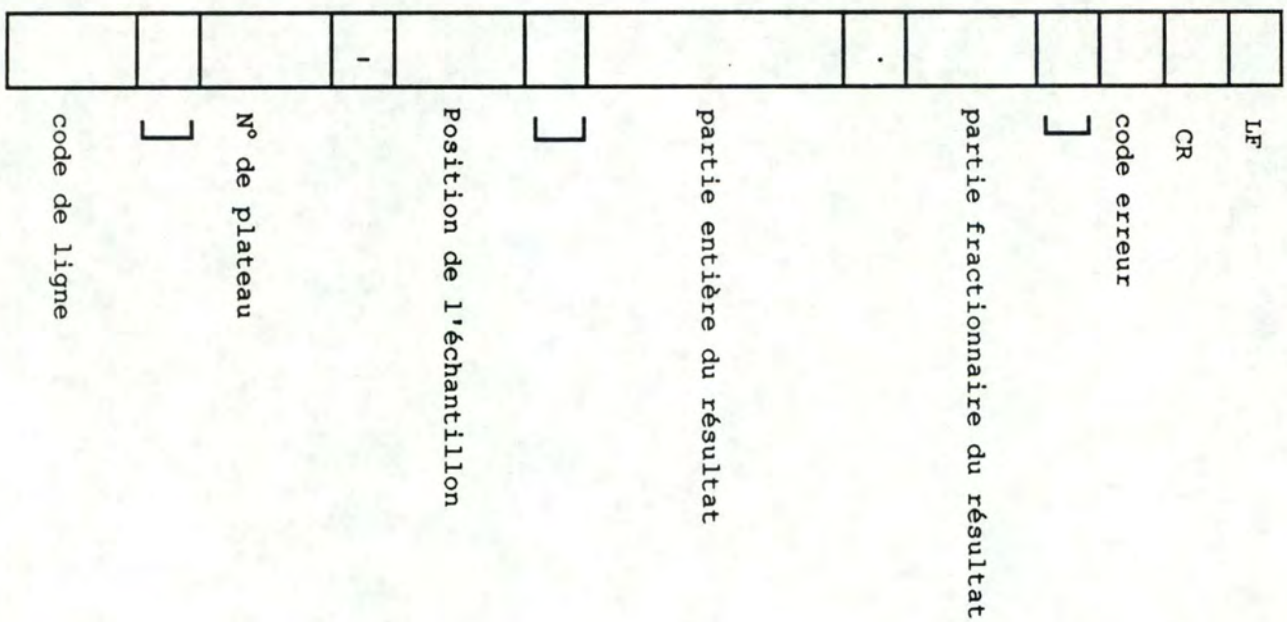
Communication

Quelques machines d'analyses proposent des interfaces pour permettre la communication avec d'autre machines ou ordinateurs. Ces interfaces peuvent être de n'importe quel type (parallèle,sériel, ...) mais une norme de fait s'impose à la lecture des spécifications de ces automates: l'interface RS232c. Dans tous les cas, ces interfaces serviront au transfert de résultats et parfois également à la programmation des machines d'analyses. Malgré la relative norme RS232c, il est interessant de remarquer que les formats de transfert utilisés par ces machines sont, eux, très hétéroclites. Ceci impose donc une programmation spécifique pour une connection éventuelle. Je vous propose ci-après les formats des résultats des machines les plus répandues sur le marché. Vous pourrez constater vous-même l'importance de ces différences.

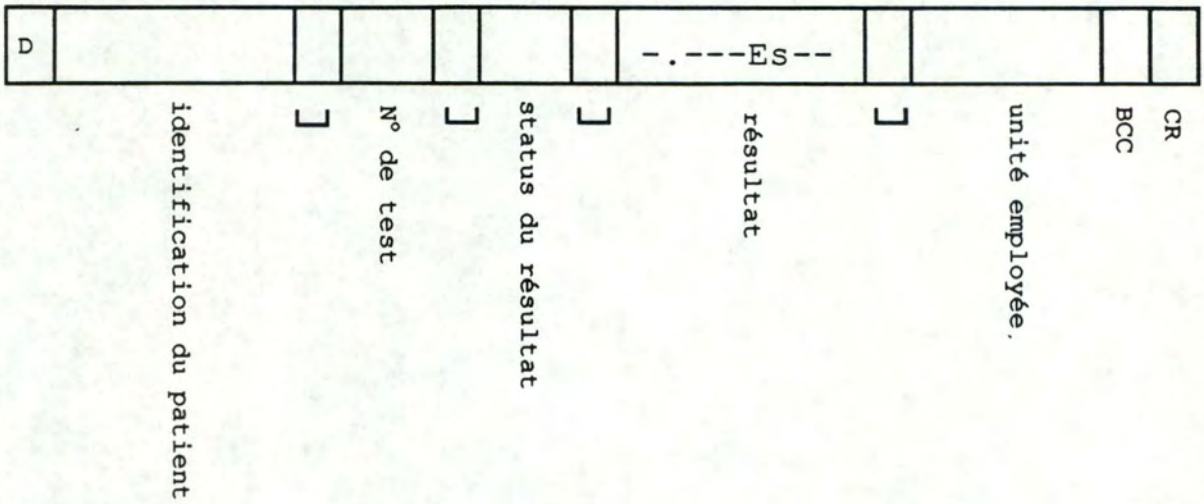
HITACHI 705



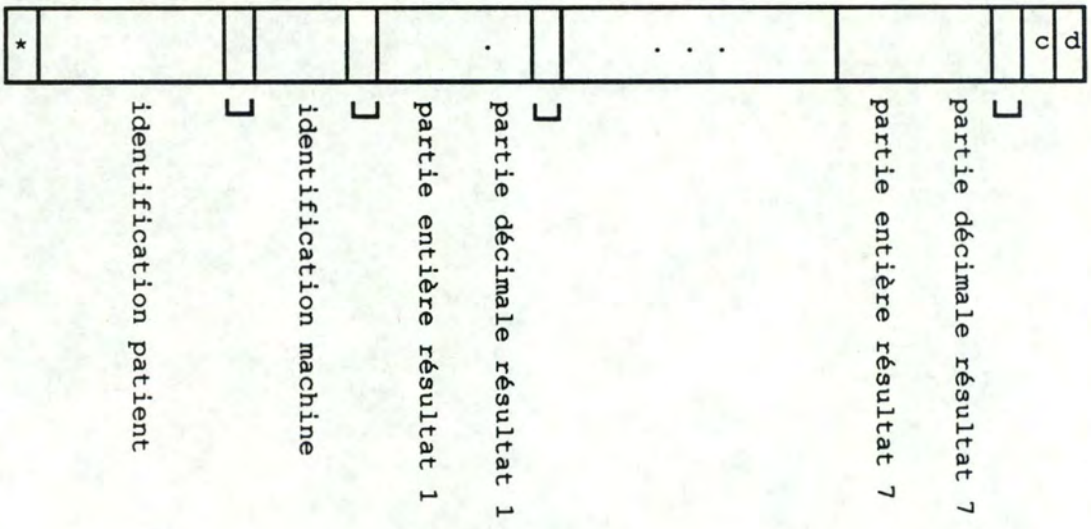
COBAS BIO



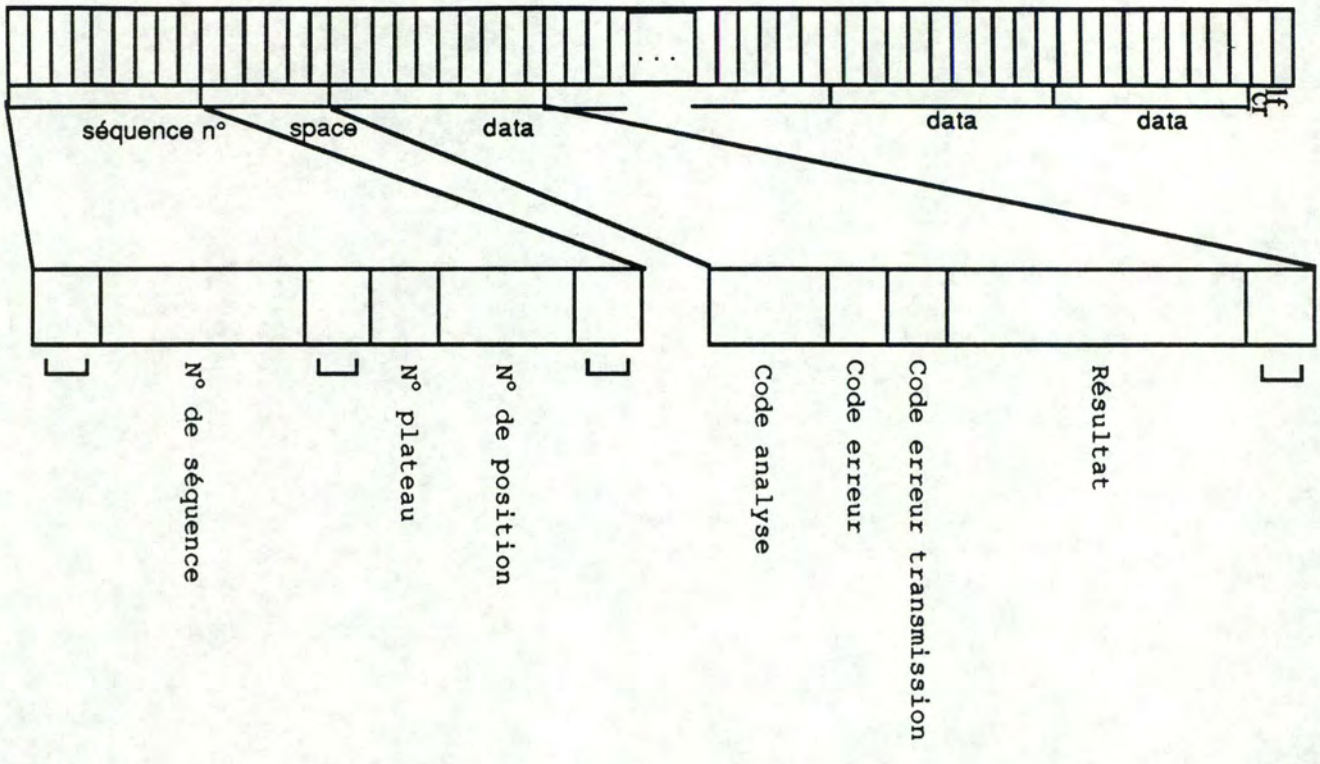
B.N.A.



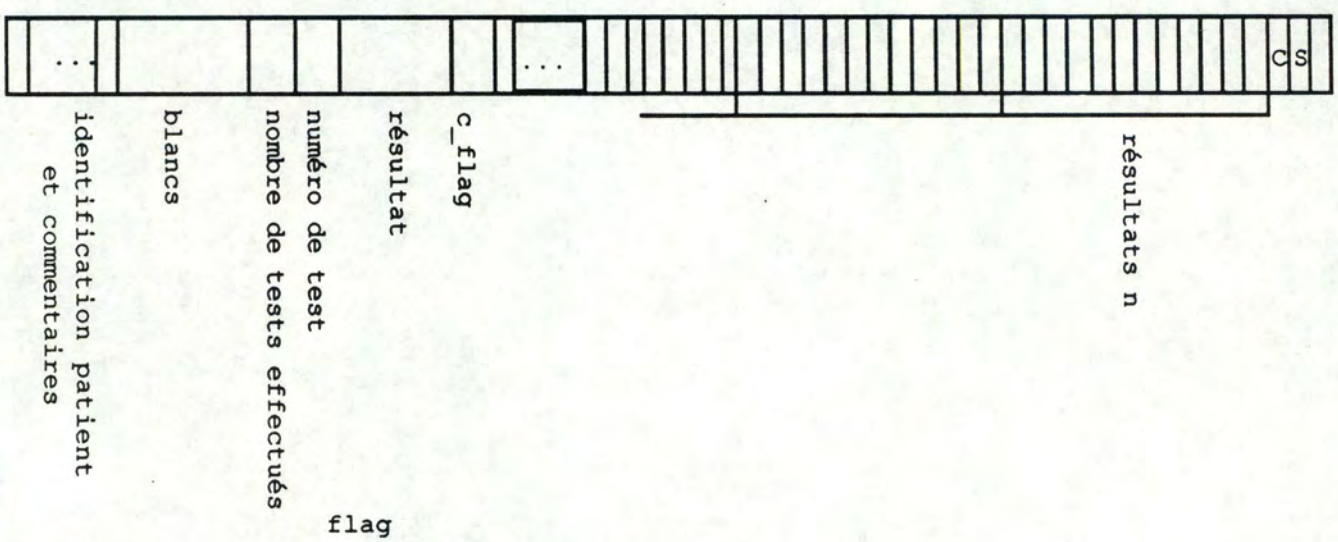
Coulter S7



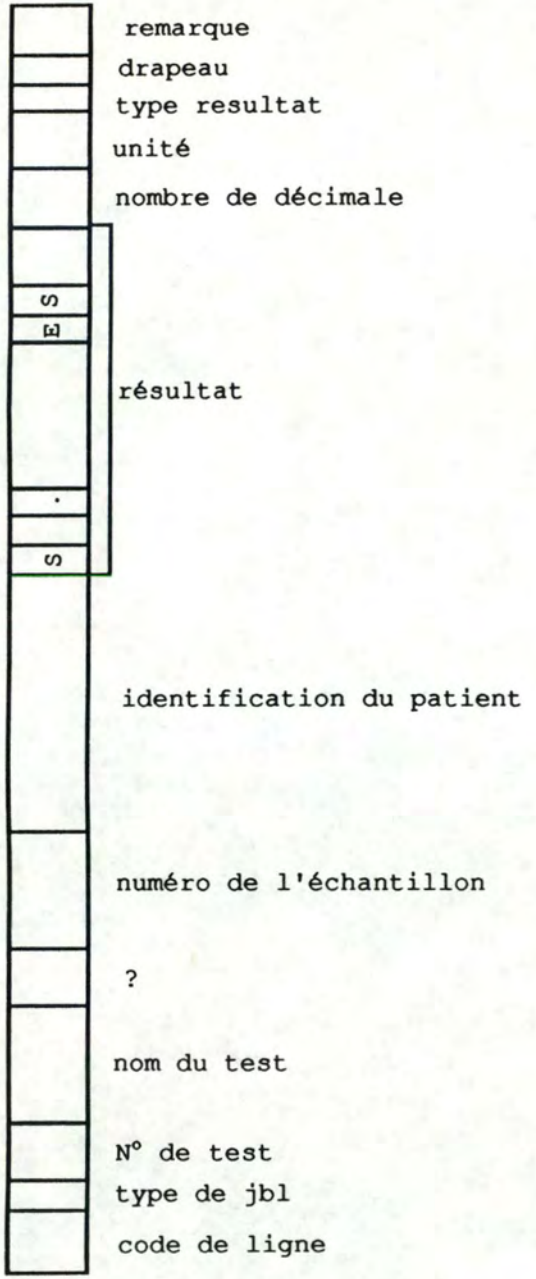
HITACHI 704



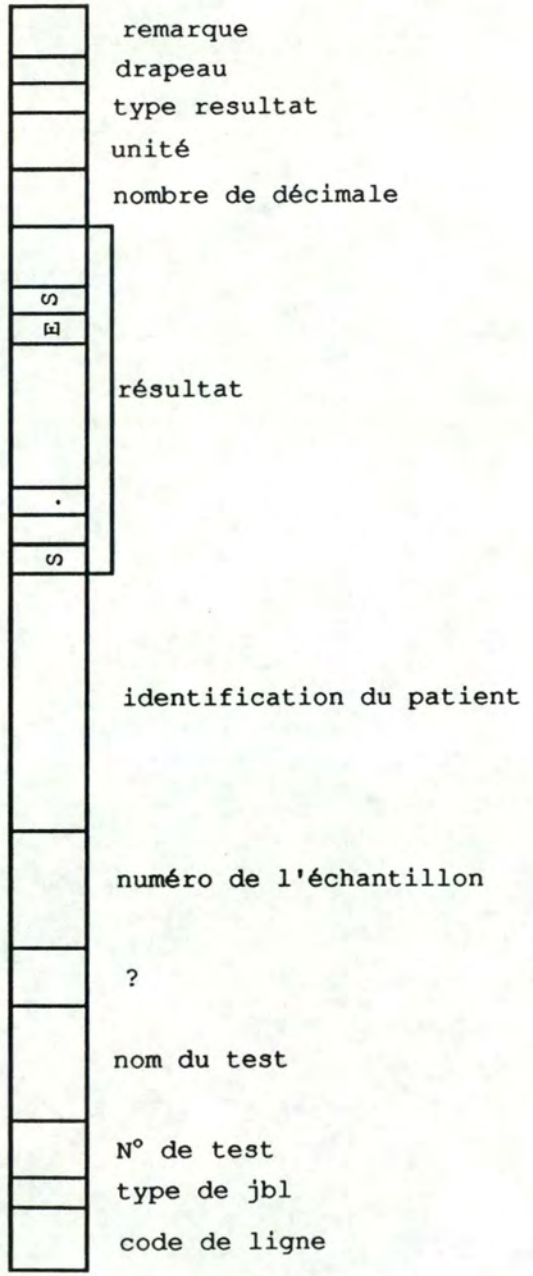
HITACHI 737



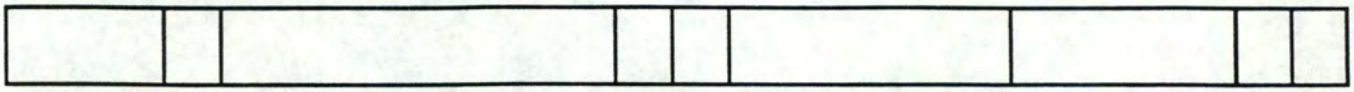
COBAS FARA



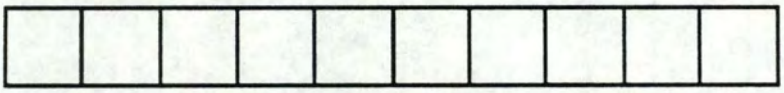
COBAS MIRA



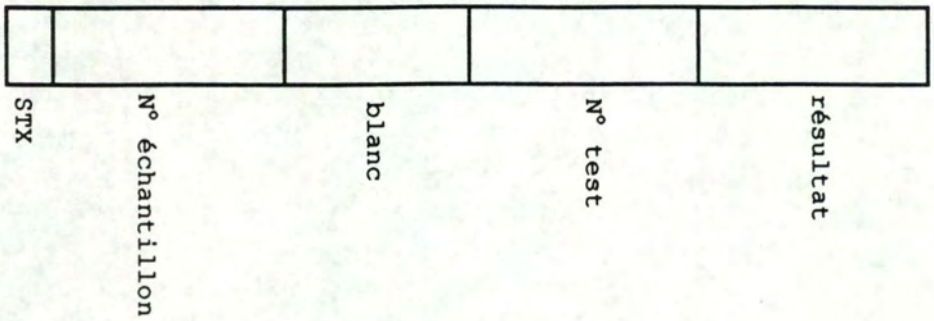
SEBIA PROFIL



MONARCH (longueur variable)



EPENDORF IRIS



Les machines d´analyses ont différentes logiques de travail

a) Réaliser toutes les analyses liées à un échantillon avant de passer à l´echantillon suivant. La machine travaille donc d´échantillon en échantillon.

b) Réaliser UNE analyse pour l´ensemble des échantillons avant de passer à l´analyse suivante; la machine, dans ce cas, travaille analyse par analyse.

c) Une troisième logique existe et est en fait l´utilisation des deux premiers principes. La machine réalise les analyses liées à un échantillon avant de passer à l´échantillon suivant SI les réactifs sont présents. Si les réactifs sont manquants, il passe à l´échantillon suivant et la machine reviendra sur ces analyses manquantes dès que les réactifs seront ajoutés.

Dans notre cas l'étude s'est portée principalement sur quatre machines

- L'HITACHI 705
- Le COULTER S7
- Le COBAS BIO
- Le BNA

Formats

Les formats de ces machines ont été donnés auparavant.

Types d'échantillons

Le sang prélevé peut être utilisé comme tel, on parle alors de sang total.

Si on laisse le sang se coaguler, un caillot se forme et une substance peut être prélevée: le sérum.

L'addition d'un anticoagulant et une centrifugation permet de récolter d'une part, les globules rouges et d'autre part, le plasma.

Les machines

1. Hitachi 705

- Les analyses effectuées sont faites principalement sur sérum mais certaines peuvent se faire sur urine
- 18 analyses différentes sont préprogrammées mais il existe la possibilité de les remplacer par d'autres.
- Le mode de travail est "patient par patient" : toutes les analyses demandées pour un patient sont effectuées avant de passer aux analyses du patient suivant. Un autre mode dit d'urgence existe et est actuellement employé pour les analyses en urgence, les analyses sur urines, les analyses sur sérum dilué et les vérifications des résultats anormaux. Ce système d'urgence ne transmet pas les résultats via le ON-LINE si bien qu'il est nécessaire de les introduire manuellement.
- Les échantillons sont placés sur les plateaux dans l'ordre d'arrivée des demandes.

2. Cobas bio

- Les analyses effectuées se font sur sérum, plasma, sérum dilué et globules rouges.
- 17 analyses sont programmées mais d´autres pourraient les remplacer.
- Le mode de travail est ´analyse par analyse´ : on effectue un type d´analyse pour 1-25,26 patients. Les analyses en urgence déterminent l´ordre des analyses accomplies sinon l´ordre est indifférent. Ceci souffre deux exceptions : a) les bilirudines totales et conjuguées sont étudiées en même temps, sur le même type d´échantillon (sérum) mais avec des réactifs différents. b) Les magnésiums céríliques et érythrocytaires sont étudiés en même temps, avec un réactif commun mais sur des échantillons différents (sérum, globules rouges). Avant le transfert des résultats, certains calculs peuvent être effectués : le magnésium erythrocytaire doit être multiplié par un facteur 5, si le résultat des bilirudines conjuguées est inférieur à 1 on transmet 1 , si le résultat des bilirudines totales est inférieur à 2, on transmet 2.
- Les échantillons sont placés sur 11 plateaux différenciés en fonction du type de l´échantillon et de l´ordre d´arrivée des demandes (Job-liste -JBL-)
- Les échantillons sont jetés quand toutes les analyses sont effectuées et qu´il y a eu confirmation du secrétariat.

3. *Coulter S7*

- Les analyses effectuées, se font sur sang total.
- Toutes les analyses sont effectuées (7) et d'autres ne peuvent pas être programmées.
- Le mode de travail est "patient par patient".
- Le coulter sort ses résultats sur une sortie parallèle BCD. Pour le rendre compatible, on utilise une interface qui sérialise les données. L'interface utilisée est la PP Printer II.
- Les échantillons sont analysés de manière séquentielle suivant l'ordre des demandes. Avant de transmettre les résultats, il est nécessaire de donner le numéro d'ordre de l'échantillon. Les vérifications se font avec un numéro d'ordre qui vaut zéro. En cas d'urgence, les résultats ne sont pas transmis, et il faut dès lors les encoder à la main.
- Les analyses sont effectuées dans l'ordre d'arrivée des demandes.

4. B.N.A.

- Les analyses se font exclusivement sur sérum.
- 16 analyses sont programmées mais d´autres pourraient l´être (70 au total); deux résultats transmis se font par calcul en fonction d´autres analyses.
- Le mode de travail est ici un patchwork entre le système "patient par patient" et "analyse par analyse". La machine travaille patient par patient, limitée aux réactifs dont elle dispose. Un patient peut être terminé en plusieurs fois.
- Les échantillons sont placés sur les ´racks´ dans l´ordre d´arrivée des demandes (JBL). Les échantillons sont jetés quand toutes les analyses sur tous les échantillons du ´rack´ sont effectuées

EIA RS232-C (OU CCITT V24)

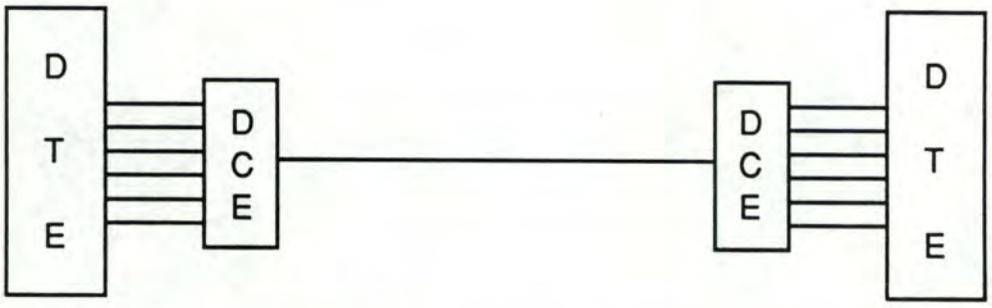
En quelques mots

Toute interface est définie en quatre points importants: les caractéristiques mécanique, électrique, fonctionnelle, et procédurale. Ces points seront abordés rapidement pour l'interface qui nous intéresse; la RS232c. Ces explications sont indispensables pour connecter deux équipements utilisant ce type d'interface mais n'est pas indispensable pour la compréhension de la suite du mémoire.

On peut dire plus ...

Normalisation

Deux organismes tentent de définir des normes et spécifications dans le domaine des télécommunications: le CCITT d'Europe et le EIA des USA. Ces deux organismes ont défini des normes fort proches pour le système le plus employé à savoir le RS232-C (EIA) et V24 (CCITT).
Ces normes sont définies pour connecter deux systèmes informatiques (DTE) distants, utilisant un système de communication (DCE) et un médium de communication (le plus employé est le circuit téléphonique commuté). Dans le cas particulier (qui est le nôtre) où les systèmes sont très près l'un de l'autre, il n'est pas nécessaire d'utiliser de DCE. Quelques modifications devront alors être apportées.



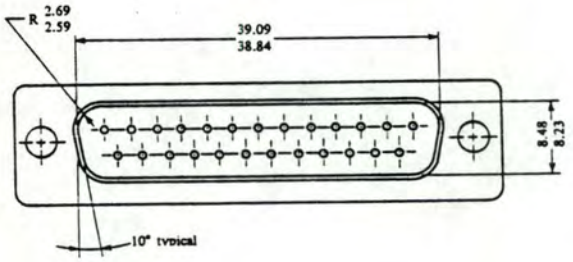
D'un côté, le DCE est responsable de la transmission et de la réception de bits via un médium de communication. De l'autre côté, le système de communication doit interagir avec le système informatique sous forme de données et d'informations de contrôle. Cette interaction ne sera possible que si quelques caractéristiques sont communes. Ces caractéristique sont de quatre ordres:

* mécanique
* électrique
* procédurale
* fonctionnelle

Interface RS232c

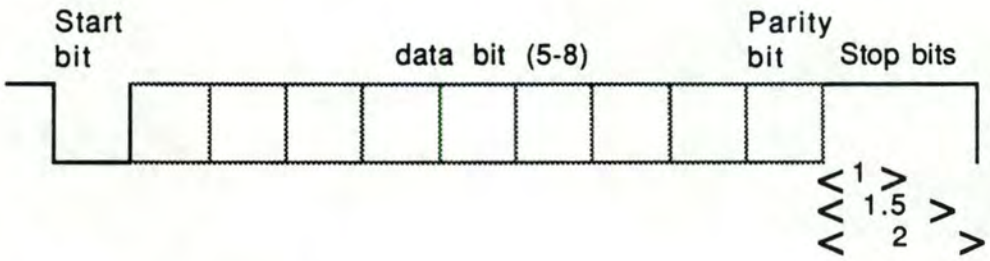
Voici les caractéristiques de ce type d'interface:

mécanique : on définit la connection physique entre les deux équipements.



électrique : on définit les tensions, les timings de changement d'état. Par convention, une tension inférieure à -3 volts est interprétée comme un 1 et une tension supérieure à 3 volts comme un 0. Les vitesses de transfert doivent être inférieures à 20 kbauds et les appareils connectés ne doivent pas être distants de plus de 15 metres

procédurale : le format des données



fonctionnelle : nous allons résumer les spécifications fonctionnelles des broches les plus importantes. Le nom des fonctions sera suivi des désignations pour les normes RS232c et V24.

Protective Ground (AA-101), cette broche doit être connectée à la terre pour la sécurité de la ligne.
Reference Ground (AB-102), cette broche donne le niveau de référence pour pouvoir juger des états des autres lignes.
Transmitted Data (BA-103), sur cette ligne seront transmises les données en provenance du DTE. Des données seront transmises uniquement si les RTS, CTS, DTR, DSR sont en état bas (1).
Received Data (BB-104), c'est par ici que seront recues les données transmises par un système distant.
Request To Send (CA-105), le DTE voudrait transmettre et il le fait savoir au DCE
Clear To Send (CB-106), en réponse à la demande de transmettre, le DCE prévient le DTE qu'il est prêt à répondre à cette demande.
Data Set Ready (CC-107), est utilisé pour indiquer que le DCE est en état de marche (Power On, pas en état de tests,...)
Data Terminal Ready (CD-108), indique que le terminal est prêt. Dans le système de connection par réseau commuté, ce signal est utilisé pour répondre et maintenir la "conversation". Si le signal s'inverse, alors la communication est coupée.
Carrier Detect (CF-109) prévient le DTE que le DCE reçoit un signal qui semble issu d'un autre DCE. Il est important de le laisser à l'état bas lors de transmission de données. S'il existe une coupure de plus de quelques millisecondes, il est remis à l'état haut et la connection est considérée comme perdue.
Ring Indicator (CE-125) lorsqu'un modem est relié au réseau téléphonique, cette broche prévient losrqu'un appel est détecté.
D'autres fonctionnalités existent mais peu d'entre elles sont couramment employées. Si vous voulez plus de précisions, il vous est loisible de vous en référer au normes définies par les organismes cités ci-avant.

Comparaison à une conversation téléphonique

Pour expliquer le scénario le plus fréquent, nous comparons la procédure de connection et de transfert à une conversation téléphonique usuelle.

1. Data Terminal Ready; vous êtes prêt à téléphoner à votre interlocuteur. vous décrochez votre téléphone.
2. Data Set Ready; votre téléphone vous donne un signal sonore : OK c'est qu'il fonctionne.
3. Request To Send; vous demandez à parler à votre interlocuteur.
4. Clear To Send; votre interlocuteur est prêt à vous écouter.
5. Transmit Data; vous parlez
6. Not Request To Send; vous avez terminé de parler : vous attendez une réponse ou vous terminez la conversation.
7. Not Clear To Send; vous coupez la conversation et votre téléphone vous envoie un signal pour vous avertir qu'il fonctionne toujours.
8. Not Data Terminal Ready; vous raccrochez votre téléphone

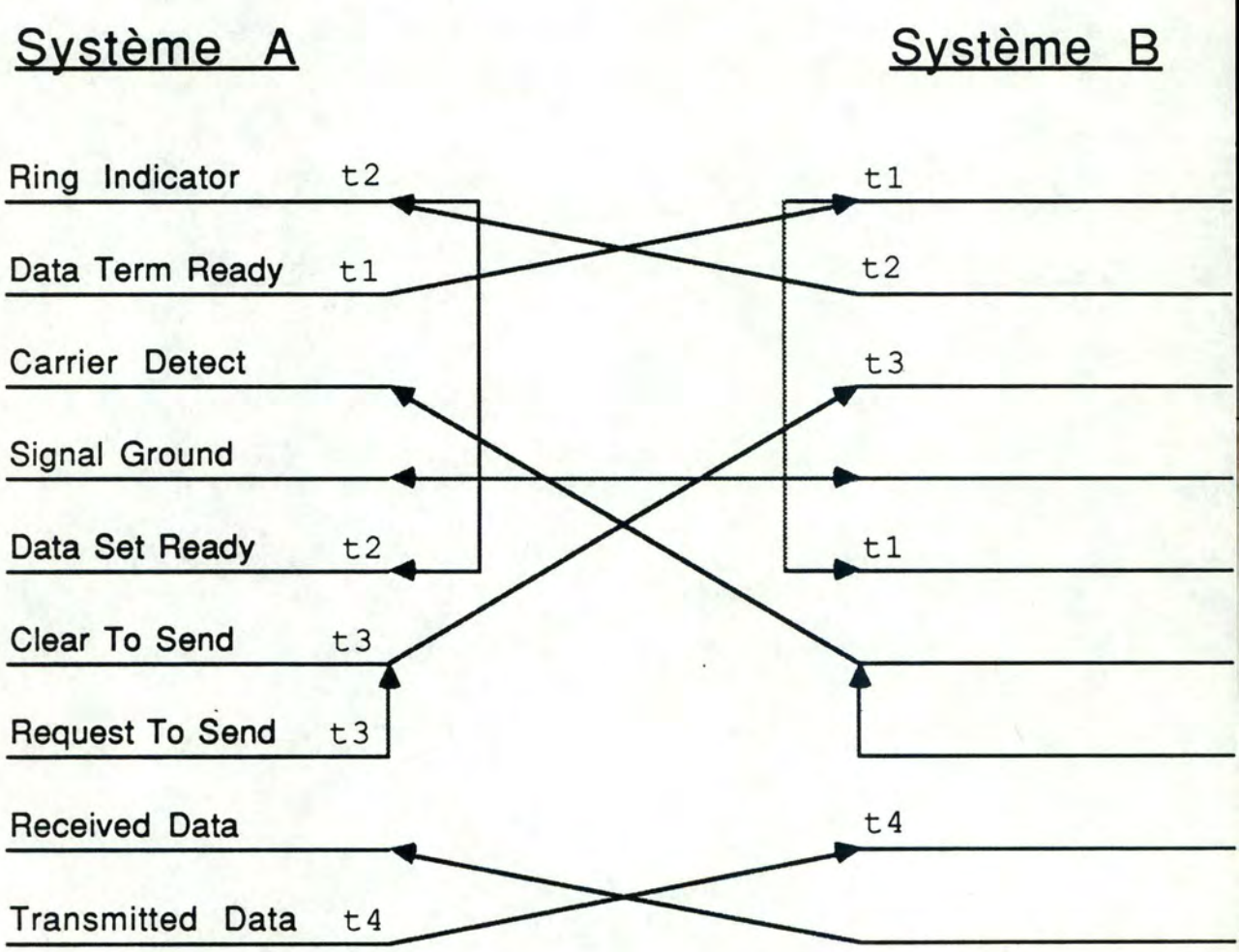
Ceci est la suite logique des signaux envoyés lors d'une "conversation téléphonique". Si vous aviez été à l'autre bout du fil, la procédure aurait été différente. La voici détaillée.

0. Ring Indicator; le téléphone sonne ...
1. Data Set Ready; vous décrochez votre téléphone et celui-ci fonctionne correctement.
2. Data Carrier Detect; vous entendez du bruit sur la ligne, on cherche à vous parler.
3. Received Data; vous écoutez ce qu'on vous dit.
4. Not Data Set Ready; votre interlocuteur a coupé la conversation.
5. Not Data Terminal Ready; vous raccrochez à votre tour.

Null-modem

Dans le cas de DTE suffisamment rapprochés, il n'est pas nécessaire d'utiliser de modem mais dans ce cas, les connections changent.
Dans le schéma qui suit, le système A cherche à transmettre des données au système B. La suite chronologique des changements d'états (t1..t4) facilite la compréhension.

Connection de DTE sans
utilisation de DCE (Null-modem)



Une telle connection utilise pourtant les broches réservées aux DCE. Il est donc nécessaire de forcer certaines de ces broches au moment opportun et en fonction d'autres broches. DTE et DCE sont ici confondus.

Cas particuliers

Jusqu'a présent nous vous avons parlé de connection à 10-16 broches. Pourtant, il existe des connections qui se font en beaucoup moins de broches: 6 et même 3 broches. Pour les liaisons 6 broches, il faut ponter les broches CTS et RTS ainsi que les broches DSR et DTR. La broche DCD peut être oubliée. Pour les terminaux, trois broches seulement sont utilisées: Reference ground, RD, TD.

Utilisation future dans notre cas

Le système RS232c pourrait être utilisé pour d'autres fonctions du logiciel du laboratoire.

* 1. Une connection avec un système informatique d'un autre laboratoire pourrait exister
* 2. Les médecins pourraient consulter les résultats de leur patients via un terminal ou un micro-ordinateur à domicile.

Le système UNINA

En quelques mots

Le système UNINA se résume en un concentrateur associé à un système de traduction de protocole. Les protocoles BSC et HDLC sont utilisés lors de transfert de données vers l'ordinateur pricipal (Host).

Mais encore ...

Base du système

Le concentrateur

L'utilisation d'un concentrateur n'est pas indispensable mais utile. En effet,

le matériel nécessaire à la connection d'une machine extérieure sur l'ordinateur central représente un coût non négligeable et la gestion des transmissions mobilise la puissance du processeur central pour des tâches auxquelles il n'est pas adapté. (traitement simple mais comportant une gestion d'intérruptions énorme)

Les lignes de transmissions représentent un certain coût, alors qu'elles ont un faible taux d'utilisation (quelques secondes par analyse)

Les caractéristiques principales :

- Mini-ordinateur dont l'architecture est orientée transmission. (set d'instructions réduit mais rapide)
- Capacité de mémoire vive quasiment inexistante.
- Système d'exploitation n'assurant que des tâches simples mais permettant de gagner le maximum de rapidité.

Les fonctions principales :

- détection et correction des erreurs sur la ligne concentrateur-host
- Désynchronisation complète des messages en provenance des interfaces machines.
- Centralisation des informations en provenance des interfaces et transfert sur la ligne concentrateur-host.

Fonction liée à la machine d´analyses.

- Possibilité d´émettre des messages des interfaces sans pour autant qu´ils proviennent du concentrateur. (fonction de pooling de demandes de résultats pour le BNA, ce qui résout le problème de l´état ´slave´ de la machine)

Une limite importante à ce sytème est la nécessité d´une ligne RS232. Certaines machines ne possèdent pas cette possibilité.

Traduction de protocoles

Cette fonction est exclusivement réalisée par les interfaces : celles-ci peuvent être programmées en fonction de la machine, du protocole, de l´OS, etc... Ceci est réalisé par la programmation d´EPROM. (mémoire morte programmable) Il est à noter que le système de transfert vers le concentrateur ne nous a pas été communiqué. Le transfert entre concentrateur et host se fait selon um protocole unique et donc indépendant de l´origine du message transmis. Plutôt que de gérer les n protocoles différents, il est possible de ne s´interesser qu´au seul protocole du concentrateur, les autres étant pris en charge par le système UNINA.

Connection machine-interface

Cette connection semble souffrir d'un manque de possibilités: il faut en effet que ce soit une connection RS232. Mais cette norme étant la plus répandue, il est presque toujours possible de se ramener à ce standard. Chaque interface peut être programmée en fonction des caractéristiques de la machine (stop bit, bits de synchro, ...) Cette programmation est réalisée entièrement par une équipe de chez UNINA. Il est donc indispensable de faire appel à leurs services pour la connection. Il m'a été dit qu'il existe actuellement sur le marché des programmes permettant de réaliser un grand nombre de connections de machines d'analyses, et en leur accordant un délai, ils pensent réaliser la connection de quasiment toutes les machines qu'on pourrait leur proposer. En plus de la connection l'interface ajoute des caractères aux messages pour permettre une reconnaissance de la provenance. Je pourrais proposer une suite explicative des normes pour l'ensemble des machines dont j'ai la documentation. Cette liste étant trop technique, je pense préférable de l'omettre.

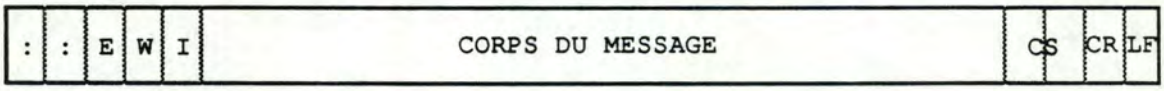
Connection concentrateur-host

Cette connection doit aussi être du type RS232C asynchrone. Il semble donc impossible d'y relier un quelconque IBM (excepté PC). Cette restriction est quasiment la seule car presque tous les terminaux utilisent cette norme. Le concentrateur est tout aussi paramétrisable que les interfaces (caractère de SOM,EOM,...). Pour réaliser la connection host-concentrateur, il faut implémenter un programme qui réalise la connection, les fonctions de vérification, de demande de transfert, etc ... (voir chapitre consacré au programme de connection) Le dialogue concentrateur-host est différent en fonction de la qualité d'unidirectionnalité ou de la bidirectionnalité. Ces deux dialogues sont expliqués sous forme de programmation de très haut niveau.

Concentrateur unidirectionnel

1. Attendre qu'il y ait un message à transmettre vers l'ordinateur host.

2. Transmettre ce message sous un format précis.



Le E représente un caractère d'erreur. Ces valeurs sont

0: normal
1: timeout
2: erreur transmission
4: message trop long
8: erreur syntaxe

Le W représente un caractère de warning. Ces valeurs sont

0: normal
1: micro-coupure de courant
2: dépassement de capacité de mémorisation de l'interface

Le I est l'identification de l'interface.
Le check-sum est obtenu en faisant la somme de tous les caractères du message, y compris ::,E,W. Cette somme est prise modulo 128 et transformée en ces deux caractères hexadécimaux.

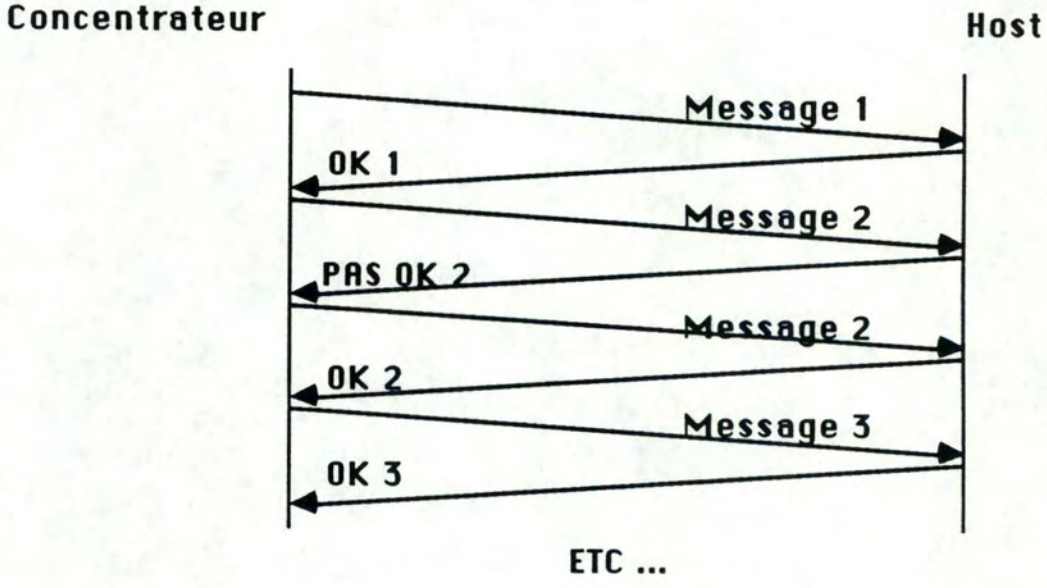
3. Le host renvoie '$$$' si le message qui vient d'être transmis est accepté. Retourner au point 1.
Si, par contre, le host renvoie '???' , le message n'est pas accepté. Aller en 2.

4. S'il existe un timeout (30 sec) alors aller en 2.

Cette manière de procéder porte le nom de BSC; tout

envoi de messages doit être accepté ou refusé avant de continuer la transmission. On obtient alors une conversation du type

B.S.C.

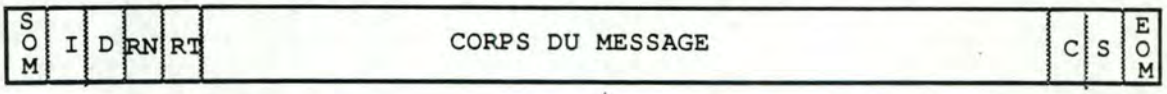


Concentrateur bidirectionnel

1. envoi de demande de transfert

2. réception de l'accord

3. transmettre les messages sous un format précis



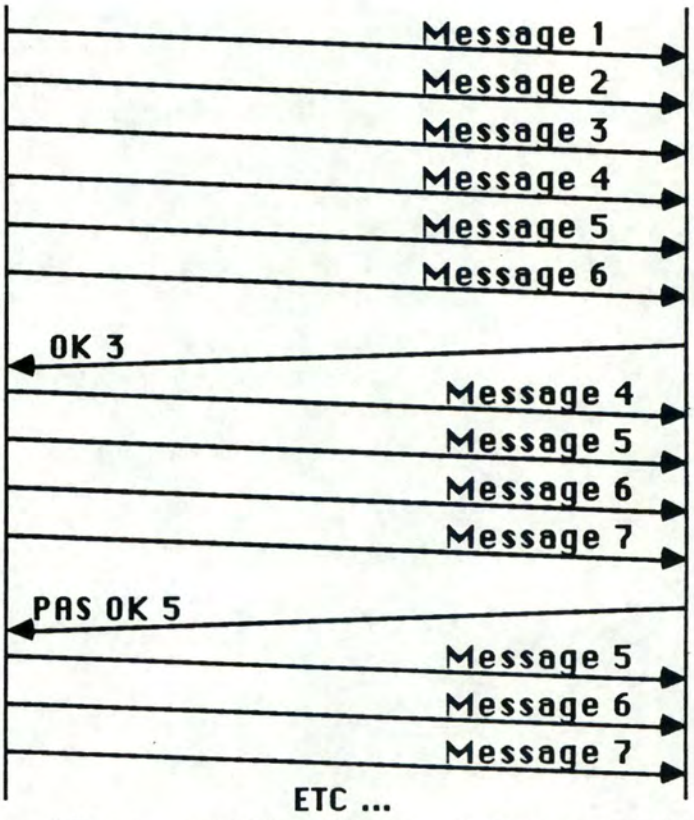
SOM : représente le début du message
EOM : fin du message
I : identification de l'interface
D : précise le type de message transmis (d=données, a=ack, n=nak,...)
RN : numéro du record ; son numéro est incrémenté à chaque transmission
RT : contrôle le découpage des messages. Si RT=0 alors il existe d'autres données pour compléter le message, si RT=1 alors le message est complet.
CHECK-SUM : obtenu en faisant la somme de tous les caractères exceptés SOM,EOM. On prend le "two's complement" de cette somme, puis le modulo 128. On doit trouver 0.

4. Le host répond par un message
   ACK N : le host accepte la transmission jusque et y compris le numéro N. Il faut donc retransmettre les messages portant les numéros N+1 et suivants.
   NAK N : le host accepte la transmission jusque et non compris le numéro N. Il faut donc retransmettre les messages N et suivants. Aller en 1.

Cette procédure porte le nom de HDLC . On peut accepter une série de transmissions en une seule fois.

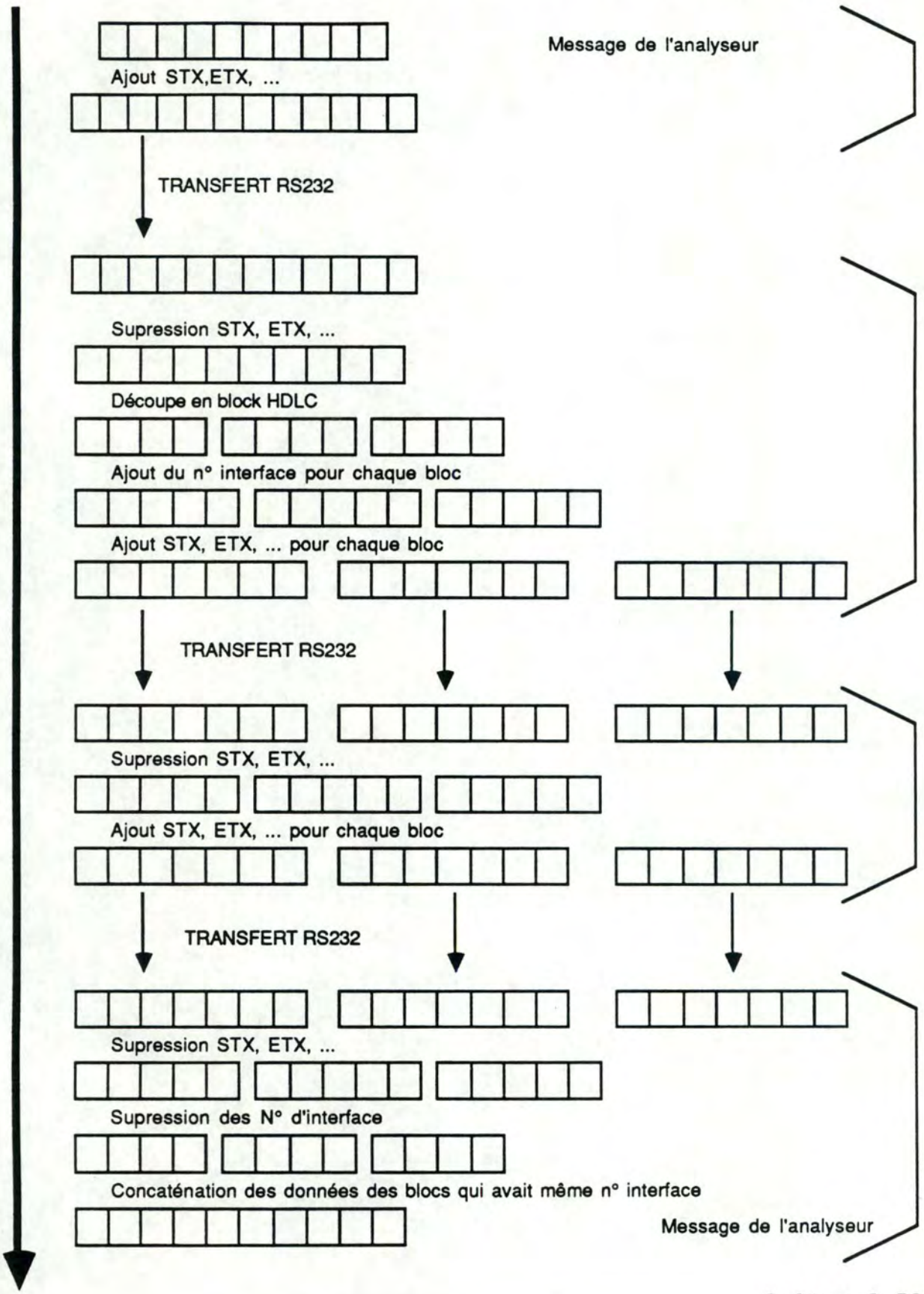
Ceci permet une plus grande rapidité et moins de trafic sur la ligne.

HDLC



L'emploi de caractères particuliers tels que SOM ou EOM peut poser un problème de transparence. Le moyen employé pour contourner cette difficulté est d'utiliser un caractère de transparence TRA. Si l'un des trois caractères SOM,EOM,TRA se trouve dans les données, on les y remplace par une suite de deux caractères avant la transmission (SOM -> TRA 1, EOM -> TRA2, TRA -> TRA3). Pour retrouver le message initial il suffit de faire le remplacement inverse après réception du message et les problèmes de transfert ont ainsi été évités.

Parcours idéal d'un message de l'analyseur



Dans notre cas

Il serait possible d'utiliser un système bidirectionnel pour le BNA et l'Hitachi. Pourtant, cette mise à jour n'a jamais été prise en considération. Une des raisons de ce choix est peut être le prix.

Les prix

Voici les prix qui m'ont été communiqués par Monsieur Demarteau, directeur de la firme.

Système unidirectionnel

concentrateur	95000	122000	selon la mémoire
interface	70000	90000	selon la mémoire
soft interface	5000	10000	

Mise à jour pour le système bidirectionnel

concentrateur	23000	pour la modification hard
interface	0	il n'existe pas de modification hard
soft interface	10000	
installation	10000	

Remarques : même si vous ne désirez mettre qu'une seule machine en mode bi-directionel, il est nécessaire de mettre TOUT le système dans ce même mode. Pour certaine machine (ex Hitachi 705), il est nécessaire d'acheter une autre interface qui est beaucoup plus coûteuse (125000)

Les quelques modalités :

payement :35% d'acompte
réalisation :15 jours
garantie soft et hard :3 mois

Dans le cadre du laboratoire, voyons le prix de la facture UNINA pour un système unidirectionnel couvrant 4 des machines. (nous prendrons les prix moyens)

1	concentrateur	108500 fr
4	interfaces	320000 fr
4	softs	30000 fr

	Total	458500 fr

La mise à jour du système pour le rendre bidirectionnel se compose comme suit.

1	MAJ concentrateur	23000 fr
4	softs interfaces	28000 fr
1	interface hitachi	125000 fr
	installation sur site	10000 fr

	Total	186000 fr

CHAPITRE II

ANALYSE FONCTIONNELLE

ANALYSE FONCTIONNELLE

Concepts de l'analyse fonctionnelle

Quels système d'information ? (s.i)

Les organisations conçoivent, réalisent et utilisent des systèmes d'information pour satisfaire les besoins en information engendrés par leurs comportements organisationnels.

Par système d'information d'une organisation nous entendons une construction formée d'ensembles:

- d'informations, représentations -partielles- de faits qui intéressent l'organisation
- de traitements, procédés d'acquisitions, de mémorisation, de recherche, de communication et de transformation des informations
- de ressources humaines, techniques et organisationnelles qui en assurent le fonctionnement.

Un système d'information n'est donc pas réduit au seul système automatisé ou informatique mais englobe des traitements automatisés, interactifs et même exclusivement manuels.

Etapes de développement d'un S.I.

Le développement d'un système d'information couvre les étapes suivantes :

- l'étude d'opportunité qui prépare un avant-projet de solution à partir des besoins exprimés par l'organisation
- l'analyse fonctionnelle qui, à partir de l'avant-projet, construit une solution détaillée et indépendante de tout moyen de réalisation
- la conception de mise en oeuvre qui affine et définit la solution retenue après la prise en considération des caractéristiques logiques des moyens de réalisation
- la réalisation et la mise au point

Etude d'opportunité

Identification du projet

Parmi les facteurs à prendre en considération pour identifier un projet d'automatisation d'un S.I. retenons :

- le relevé des causes profondes d'insatisfaction du S.I. existant
- la localisation des déficiences

Définition du projet cadre : spécification des besoins

Cette étape constitue l'acte fondamental d'un projet informatique. Il contient la spécification des besoins exprimés sous forme d'objectifs à atteindre. Ceux-ci résultent de l'analyse des causes profondes d'insatisfaction, qui conduisent les responsables d'une organisation à envisager des modifications du S.I. existant.
On décrira les contraintes liées au développement du projet (réglementaires, d'organisation, de réalisation,...) On identifiera ensuite les activités de l'organisation directement associées à la réalisation des objectifs. On fixera finalement les critères et les contraintes en fonction desquelles on veut réaliser ces objectifs :

- critères d'efficacité informationnelle : concernant les qualités des informations produites par le S.I. et les caractéristiques opérationnelles des processus du S.I.
- critères d'efficacité organisationnelle : visant à mesurer l'impact que le changement induit par une nouvelle solution pourrait avoir sur le comportement des individus et des groupes : amélioration de la communication au sein de l'organisation
  * satisfaction et motivation des individus face à leur travail

* capacité d'adaptation des idividus à une nouvelles technologie
* modifications de l'emploi et réactions face à celles-ci
* répartition du pouvoir au sein de l'organisation

- critères d'efficacité économique : portent essentiellement sur les coûts et les économies d'un nouveau système. Les objectifs en matière de coûts représentent des contraintes budgétaires d'investissement ou de fonctionnement : nature des dépenses (personnel, équipement, logiciel, ...) et des activités (implémentation, maintenance, exploitation, ...)
- critères d'efficacité de réalisation : durée de vie du projet et délais de mise en exploitation

Etude critique du S.I. existant

Il est généralement nécessaire de fonder ou de justifier les éléments d'un projet cadre par une analyse du S.I. existant. L'étude du système existant se fera à partir d'un outil constitué de diagrammes de flux.
La critique du S.I. existant consiste en

- une critique fonctionnelle qui est une évaluation en termes des efficacités informationnelles, organisationnelles et économiques
- une critique structurelle, relative à l'organisation du S.I. existant, et abordant le contenu et la destination des messages ainsi que la réalisation des traitements.

Elaboration de solutions

Il faut prendre en considération quatre dimensions

- la definition fonctionnelle des traitements
- la technologie
- les personnes
- les structures d'organisation

Elaborer une solution consiste à redéfinir les activités et les règles de traitement de l'information, à envisager des modifications technologiques, à redéfinir les fonctions et les rôles assumés par les personnes et à repenser les structures d'organisation

Normalement, une solution devrait être décrite en termes généraux. L'essentiel de la solution est exprimé par un diagramme de flux construit à partir des mêmes règles que celles adoptées pour l'étude de l'existant.

- la phase représentera le niveau de décomposition des traitements
- la référence à la mémoire du S.I. est faite en termes de collections d'informations.

Le diagramme de flux est complété par les spécifications des éléments suivants :

- les objectifs associés à une phase et les performances attendues de celle-ci
- la nature des ressources à mettre en oeuvre au niveau de chaque phase
- les principales modifications de fonctions de l'organisation et des rôles assumés par les personnes.

Analyse conceptuelle

On distinguera deux étapes majeures dans l'analyse conceptuelle :

- l'élaboration et la validation des sous-schémas conceptuels pour chaque phase du projet comprenant des opérations de traitement de l'information partiellement ou totalement automatisables
- l'élaboration et la validation du schéma conceptuel global par consolidation des sous-schémas conceptuels relatifs aux phases du projet

Nous nous limiterons d'une part à l'élaboration d'un sous-schéma conceptuel des traitements avec

- décomposition de chaque phase en fonction
- spécification de la statique d'une phase

et d'autre part à l'élaboration de sous-schémas conceptuels des informations, constitués par des schémas entités/associations

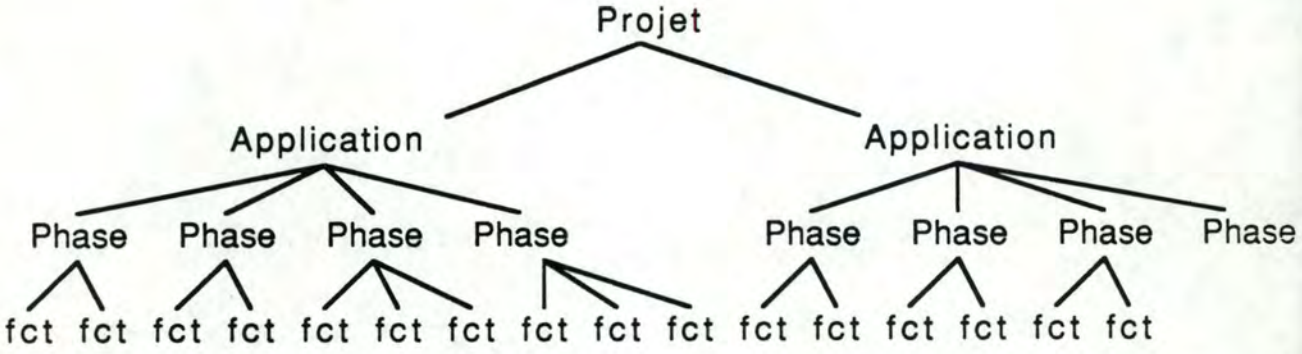
Dans le travail de description de chaque fonction de la phase, on veillera à spécifier parmi les actions primitives de traitement

- les consultations de la mémoire du S.I. en termes d'attributs d'objets
- les ajouts ou suppressions
- les modifications

Modèle de la structuration des traitements

Le modèle de la structuration des traitement est une décomposition basée sur les propositions suivantes:

- tout traitement est décomposé sous forme arborescente: un traitement de niveau intermédiaire provient de la décomposition d´un seul traitement de niveau supérieur et se décompose en n traitements de niveau inférieur.
- la décomposition d´un objet en traitements de plus en plus élémentaires correspond à une démarche par raffinements successifs
- tout traitement est décomposable en un nombre quelconque de niveaux
- on distinguera des niveaux ou repères privilégiés, en une nomenclature standard :



Définition des différents niveaux de l´arborescence

Le projet

Un projet est la partie du S.I. qui fait l´objet d´une analyse.

L´application

Une application est un traitement quasi autonome par rapport aux autres applications du projet. Une application est en communication avec d´autres par l´échange d´agrégats d´informations se faisant ponctuellement.

La phase

Une phase est un traitement possédant une unité spatio-temporelle d´exécution et est exécutée dans une cellule d´activités. Elle est le lieu

- d´identification de structures homogènes de données et de traitements, de définition d´allocation des ressources humaines, matérielles ou logicielles nécessaires à son exécution
- de redéfinition des tâches, fonctions, responsabilités et postes de travail

La fonction

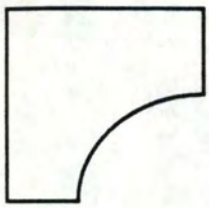
Une fonction correspond au niveau élémentaire de la nomenclature des traitements, elle correspond à la décomposition d´une phase en sous-problèmes, elle spécifie les règles de traitements relatives à la production de messages et aux actions sur la mémoire du S.I. exprimées sous forme de primitives de traitement. Elle possède une sémantique simple.

Diagramme de flux d'information

Un diagramme des flux constitue un outil simple, apte à représenter le fonctionnement, non seulement d'un S.I. existant mais aussi d'un S.I. projeté :

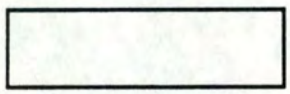
- il décrit le cheminement des messages, leurs origines et leurs destinations
- il fournit une localisation spatiale des points de naissance, de transformation et de disparition des informations. Un des axes permet de localiser les lieux de l'organisation (poste de travail, services, ...) où se développent les flux tandis que l'autre est associé au cheminement des messages
- pour chaque traitement, représenté par un rectangle, on indique les messages et les informations mémorisées en entrée et en sortie. Pour un message, on peut indiquer le nombre d'exemplaires créés. Les informations mémorisées sont représentées par un cylindre. Les flux des messages sont représentés par des flèches. Sur celles-ci, on peut représenter le nombre d'exemplaires transmis d'un message donné.

A ce stade on pourra définir les traitements comme étant des phases.

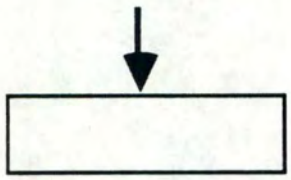


Message

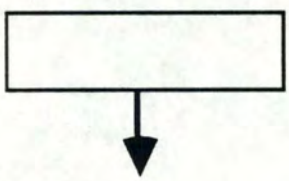
Fichier



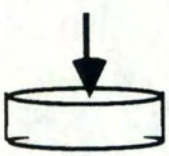
Traitement



Réception d'un message en X exemplaires



Production d'un message en X exemplaires



Opération de création / modification d'un fichier

Opération de consultation d'un fichier

Modèle de structuration des informations

Le modèle entité - association propose de modéliser les informations du réel perçu à partir de trois concepts élémentaires:

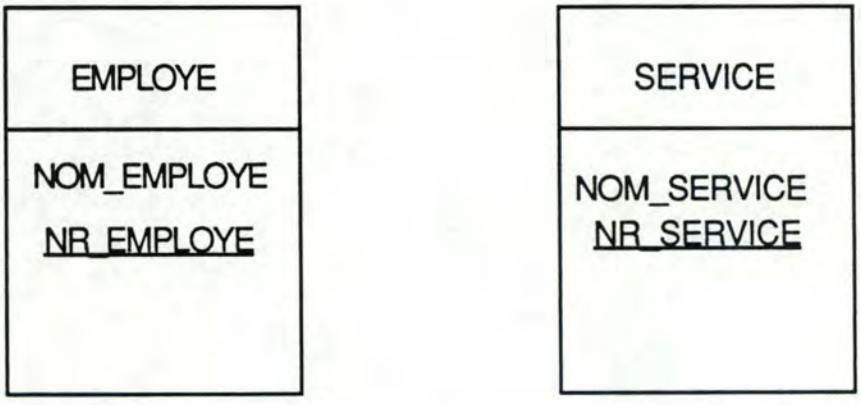
- l´entité
- l´association
- l´attribut

L´entité

L´entité est une chose concrète ou abstraite appartenant au réel perçu à propos de laquelle on veut enregistrer des informations. Une entité n´existe en tant que telle que par rapport à un individu ou un groupe qui la considère comme un tout, lui confère une existence autonome et la distingue d´autres entités et de son environement. Une entité peut posséder des attributs.

Très généralement, on ne s´intéresse pas à chaque élement individuel mais à des classe d´éléments. En parlant de patient on désignera par exemple toute personne pour laquelle est effectuée au laboratoire une analyse quelconque. On dira que patient est un type d´entité.

Un type d´entité est la classe de toutes les entités possibles du réel perçu qui vérifient la définition constitutive du type.

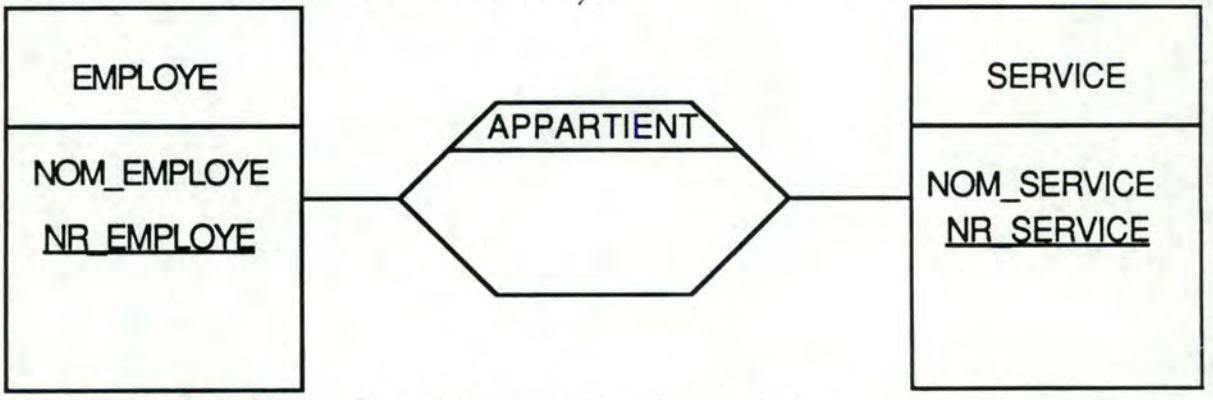


L´association

Une association est définie par une correspondance entre deux ou plusieurs entités où chacune assume un rôle donné. On veut enregistrer de l´information relative à cette correspondance. Une association peut posséder un ou plusieurs attributs. L´existence d´une association est contingente à l´existence des entités qu´elle met en correspondance.

Comme pour l´entité, on aura un type d´association, une occurence d´association.

Un type d´association est la classe de toutes les associations possibles du réel perçu qui vérifient la définition constitutive du type.

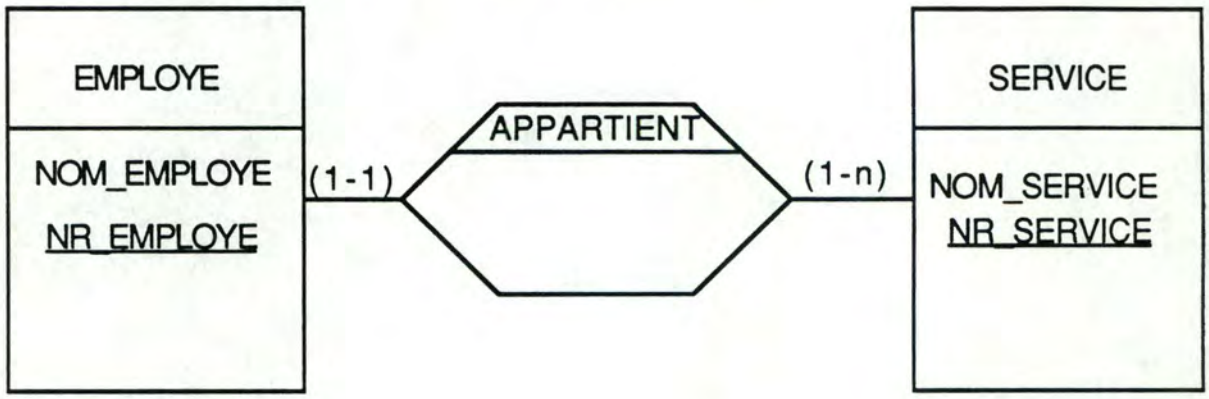


Les propriétés d´un type d´association sont :

- existence : Soit R(E1,E2,...,EN) un type d´association défini sur les types d´entités E1,E2,...En. Si une contrainte d´existence est affirmée pour Ei (1<i<n) alors toute occurrence de Ei doit à tout moment participer à au moins une occurrence de R.

  Exemple : une occurrence de patient peut exister dans le système d´information sans pour autant faire partie d´une occurrence de l´association réalisation reliant patient et analyse. Par contre, toute occurrence de mutuelle à tout moment doit être reliée à une occurrence de affiliation.

- multiplicité d'occurrences : cette propriété indique le nombre maximum d'occurrences de R auxquelles une occurrence de Ei peut participer
- connectivité : les deux propriétés précédentes peuvent être combinées en une seule propriété ; la connectivité. Soit R(E1,E2,...,En) un type d'association défini sur les types d'entités E1,E2,...En. La connectivité de R est définie par un ensemble de couples d'entiers (Min i, Max i) où
Min i indique le nombre minimum d'occurrences de R auxquelles toute occurrence de Ei doit participer à tout moment
Max i indique le nombre maximum possible d'occurrences de R pour toute occurrence de Ei



L'attribut

Un attribut est une caractéristique ou qualité d'une entité ou d'une association

UTILISATION DE L'ANALYSE FONCTIONNELLE

ANALYSE D'OPPORTUNITE

Identification du projet

Spécification des causes profondes d'insatisfaction

Les causes d'insatisfaction sont nombreuses et il serait laborieux de les énumérer de manière exhaustive. Quelques-une pourtant semblaient importantes puisque ce sont celles-ci qui m'ont permis de faire mon stage au laboratoire. Ces causes sont :

- le temps nécessaire à l'encodage des résultats est long. En effet de nombreux résultats sont introduits à la main.
- l'heure à laquelle peut commencer cet encodage est relativement tardive. Les résultats des analyses ne sont transmis que lorsque toute les analyses d'un JBL sont terminées. Or ces résultats arrivent en fin de journée uniquement.
- les temps de réponses aux médecins étaient de 48 heures, ce qui semble être trop long sur le marché. La plupart des laboratoires proposent les résultats dans les 24 heures. Il était donc nécessaire de faire les modifications nécessaires dans le but de s'aligner sur la concurrence.
- il semblait possible de réduire le travail des encodeuses en les libérant de certaines tâches que les laborantins auraient pu faire eux-même.
- la fiabilité des résultats pourrait être meilleure si les laborantins s'occupaient eux-même de certaines vérifications

Remarquons que toutes ces insatisfactions concernent un projet bien réel : la mise ON-LINE de machines d'analyses. La suite de l'analyse se fera sur les possibilités de mise ONLINE.

Localisation des déficiences

Les déficiences se situent exclusivement dans deux services:

- le secrétariat qui encode les résultats. Ne serait il pas possible de rendre l'encodage plus rapide encore ?
- le laboratoire. Ne pourrait-on pas commencer les analyses plus tôt ou encore réduire le temps des analyses ?

Définition du projet cadre

Spécification des objectifs de l'organisation

Les causes d'insatisfactions reprises plus haut représentent en fait les spécification des objectifs de l'organisation.

Spécification des objectifs informationnels

Précision des résultats

La précision des résultats est un des objectifs du laboratoire. Or il est fréquent que des erreurs se glissent dans la recopie ou à l'encodage. Ces erreurs peuvent avoir des conséquences catastrophiques et il convient de les limiter au maximum.

Délais de traitement

Il est important que les performances du système de traitement de l'information soient bonnes. Les délais de traitements doivent être aussi courts que possible.

Fiabilité

La fiabilité d'un processeur sera quasiment irréprochable pour les mêmes raisons évoquées ci-avant concernant la précision de l'information..

Durabilité

N'oublions pas que ce système de traitement de l'information est censé être vendu et il sera important que la durabilité (extensibilité du système, possibilité de maintenance, portabilité du logiciel) de celui-ci soit le plus étendue possible.

Spécification des activités concernées

Seule la phase encodage des résultats peut être modifiée. Dans certain cas, en effet, l'encodage sera remplacé par un système de vérification et de transfert de résultats de machines automatiques.

Spécification des critères d'efficacité

On cherche à se définir une série de critères et de contraintes qui serviront plus tard de critère d'évaluation et aussi de points de référence pour le contrôle de la mise en oeuvre et de l'exploitation du projet. Pour chaque critère nous allons exprimer nos préférences en trois zones de valeurs : idéale, acceptable, inacceptable.

Critères d'efficacité informationnelle

Précision des résultats

La précision du résultat est très importante et pourtant des erreurs se glissent.

+ zone idéale : 0%
+ zone acceptable : 2%
+ zone inacceptable : 5%

Délais de traitement

Les délais de traitements doivent être comparés au système actuel d'encodage.

+ zone idéale : 10 * plus rapide
+ zone acceptable : 2 * plus rapide
+ zone inacceptable : pas plus rapide

Fiabilité

La fiabilité d'un processeur ne devrait pouvoir être mise en doute. Et pourtant ...

+ zone idéale : 0% d'erreurs
+ zone acceptable : 1% d'erreurs
+ zone inacceptable : 2% d'erreurs

Durabilité

En ce qui concerne la durabilité du système il me semble très difficile d'exprimer ses préférences. Tout ce que nous pouvons exiger de ce système est qu'il soit le plus indépendant possible des machines d'analyses.

Critères d'efficacité organisationnelle

Transfert de travail

A priori, les impacts ne sont pas nombreux. Un surcroît de travail et un effort de formation va être demandé aux laborantins et ceci au profit des encodeuses.

Problème de communication

Un problème de communication va se trouver résolu. Lorsqu'une erreur apparaissait dans les JBL, il était nécessaire que les laborantins expliquent au secrétariat la marche à suivre pour la corriger. Ces palabres pouvaient durer longtemps et le résultat n'était pas assuré. Grâce à ce nouveau système, les laborantins pourront, dans une certaine mesure, corriger eux-même les erreurs. (reprogrammation y comprise)

Programmation des machines d'analyses

Actuellement la programmation de la machine d'analyses semble redondant avec la programmation des JBL. Ne serait il pas possible de supprimer cette redondance?

Critères d'efficacité économique

Heures supplémentaires

En répartissant ainsi les tâches et en modifiant quelques points organisationnels, il est possible de limiter les heures supplémentaires aux coûts trop élevés (Rem: actuellement, les heures supplémentaires prestées ne sont pas payées mais une mauvaise humeur s'est installée).

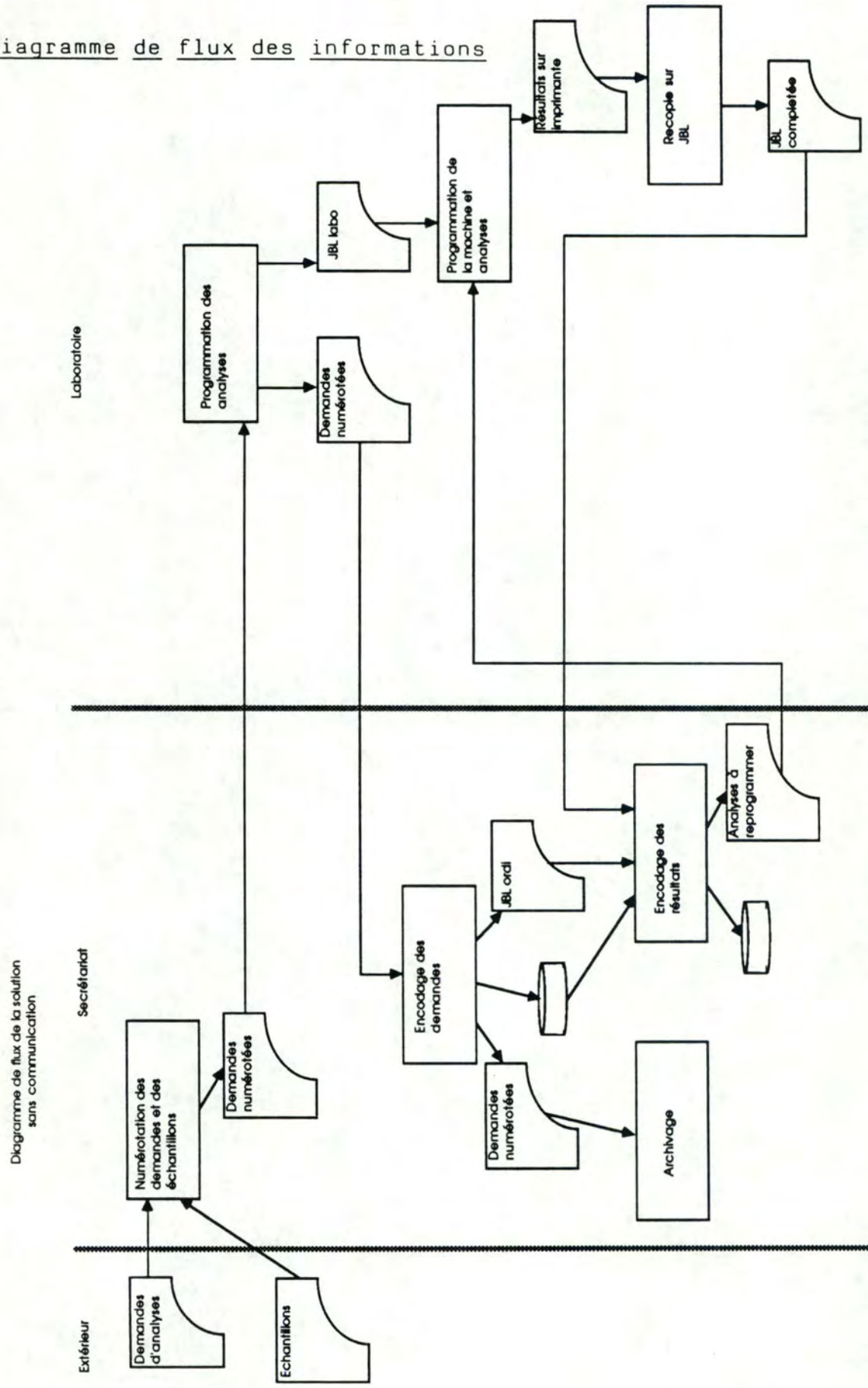
+ zone idéale : sans heures supplémentaires
+ zone acceptable : minimum d'heures
+ zone inacceptable : sans changement par rapport à la solution précédente

Critères d'efficacité de réalisation

Les délais de mise en oeuvre semblent devoir être réduits au minimum.

+ zone idéale : début janvier
+ zone acceptable : début mars
+ zone inacceptable : debut mai

Diagramme de flux des informations



Description succincte du diagrame de flux

Ce diagramme s'intéresse principalement aux problèmes liés aux machines d'analyses susceptibles d'être connectées.
Les échantillons et les demandes d'analyses sont acheminés au secrétariat qui les numérote. Ces échantillons et ces demandes sont alors acheminés au laboratoire où un laborantin va les programmer sur joblistes labo puis celles-ci seront acheminées aux spécialistes des machines automatiques qui va proceder à la programmation des machines ainsi qu'au placement des échantillons. Les analyses peuvent alors débuter. Pendant ce temps, les demandes retournent au secrétariat qui les encode. Les résultats des analyses des machines automatiques sont alors transmis sur imprimante (ou terminal). Les JBL labo sont alors complètées par les résultats nécessaires. Quand les analyses sont terminées et les JBL ordinateur sortie (top horaire ou fin d'encodage) on peut s'intéresser à l'encodage des résultats. Lorsque l'encodage est terminé, les protocoles sont sortis et envoyés aux médecins.

Compléments statistique.

- Le nombre de demandes d'analyses varie de 80 à 120 par jour
- Le nombre d'analyses par demande varie énormément; une moyenne doit se situer à plus ou moins 90 analyses.
- Durée moyenne pour chaque traitement. (Les chiffres proposés sont peut être un peu surfaits car les employés savaient que je les chronométrais. Ces chiffres sont basés sur 40 demandes d'analyses.)
  * numérotation des échantillons : 15-20 minutes.
  * programmation des analyses : 30-35 minutes.
  * encodage des demandes et des signalitiques patient : 60-70 minutes
  * programmation des machines d'analyses : 0-15 minutes (en fonction des machines)
  * recopie sur JBL labo : 15-25 minutes (en fonction des machines)
  * encodage des résultats : ???
  * reprogrammation : 2% sur le total des analyses

Critique du S.I. existant

Goulot d'étranglement

* L'encodage des résultats n'est possible qu'en fin de journée. Or cet encodage prend pas mal de temps. De plus, le laboratoire cherche à envoyer les résultats dans la journée. Il existe à ce niveau un goulot d'étranglement.
JBL Labo = JBL ordi
* Les JBL labo et les JBL ordinateur sont crées de deux manières différentes mais ces deux JBL devraient être identiques. Il serait possible de n'en faire qu'une mais alors la vérification liée à cette double programmation n'existerait plus.

Elaboration de solutions

Possibilités

Nouveautés

Les technologies nouvelles se succèdent sur le marché, on vous propose de nouveaux automates d'analyses médicales plus performants encore, de nouvelles machines qui permettent de rendre automatique des analyses qui jusqu'à présent étaient manuelles, ou encore des 'kits' qui permettent une meilleure rentabilité d'analyses manuelles. Toutes ces solutions sont valables mais ô combien coûteuses.

Connections

Une autre solution est de chercher à éviter l'encodage des résultats. Dans la présentation des machines d'analyses il vous a été présenté une caractéristique commune à la plupart de ces automates : l'interface de communication. En utilisant cette interface, il est possible d'une part de transmettre les résultats directement à l'ordinateur principal et ainsi d'éviter le travail d'encodage des résultats transmis par ces machines; d'autre part il est possible de "programmer" la machine d'analyses en fonction des analyses demandées et ainsi d'éviter le travail de programmation des machines d'analyses.

Diagramme de flux

Ci-après, vous pourrez trouver les diagrammes de flux de ces deux possibilités.

Diagramme de flux d'une solution à base de communication bi-directionnelle

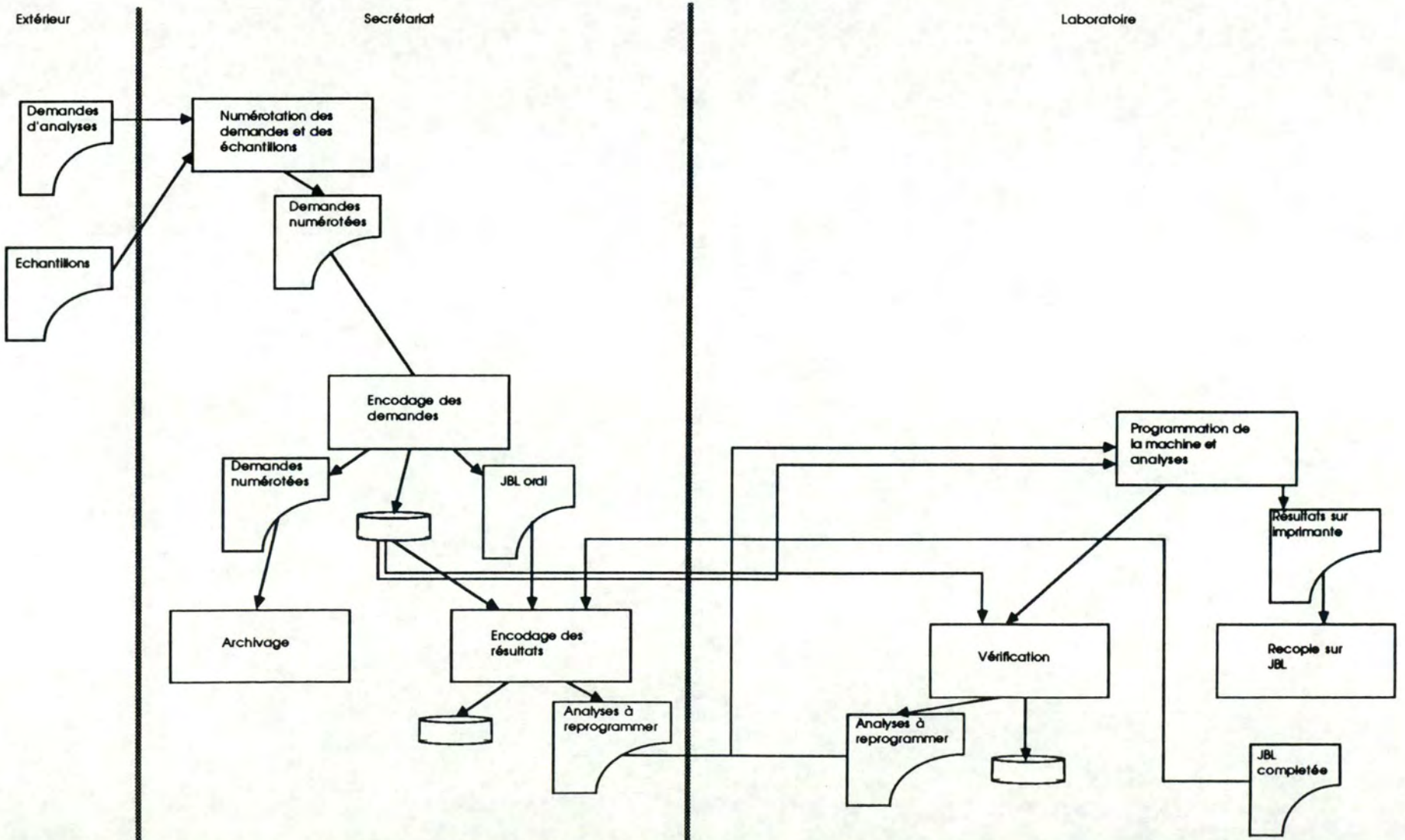
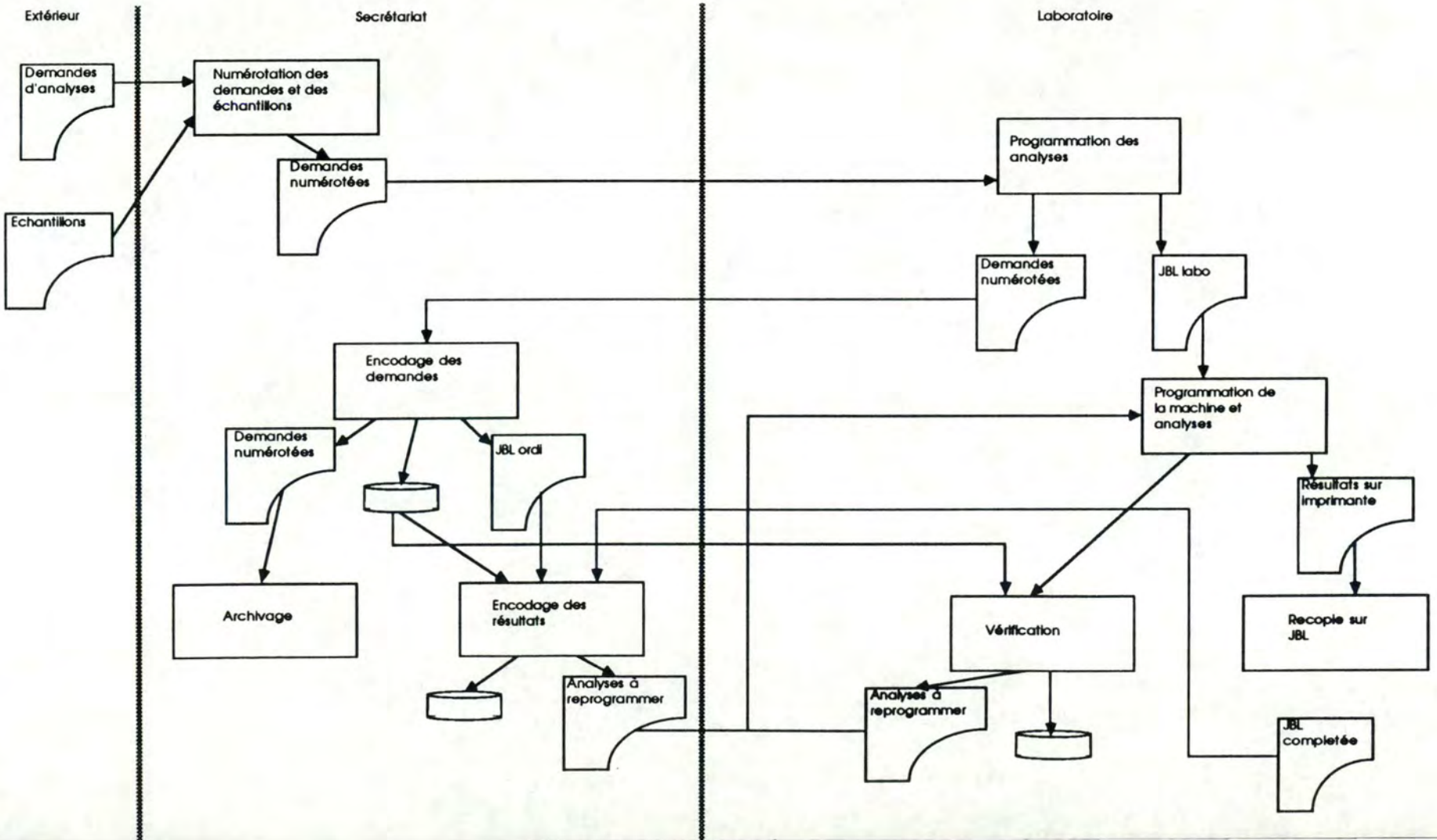


Diagramme de flux d'une solution à base de communication uni-directionnelle



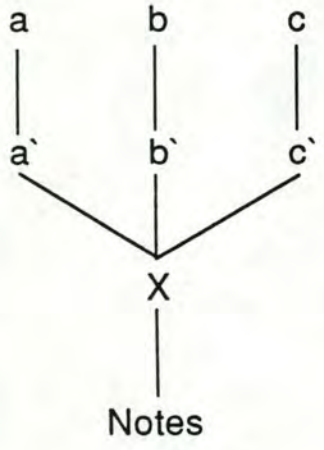
Il est évident que pour être rentable, cette solution suppose un nombre important d'analyses effectuées par ces automates; de plus il est impossible de ne faire aucune vérification sur les résultats transférés. Le travail d'encodage est alors remplacé par une simple vérification.

Solutions technologiques

Comparaison

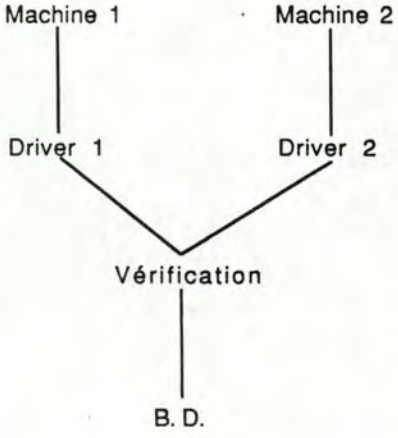
Pour faciliter la compréhension de la solution technologique, une comparaison à une conversation multilingue vous est proposée.

Conversation multilingue



Plusieurs interlocuteurs (a,b,c) cherchent à entrer en contact avec Monsieur X. Ces interlocuteurs (a,b,c) parlent des langues différentes et inconnues par Monsieur X. Pour se comprendre, il doivent faire appel à des traducteurs (a´,b´,c´) qui lors de la traduction prennent en compte les expressions consacrées telles "Tomber dans les pommes", etc... . Monsieur X peut prendre des notes qui seront dans un format qui lui est propre (sténo,...)

Comparaison à notre problème



Considérons que les interlocuteurs soient les machines d´analyses. Celle-ci transmettent leur résultats qui sont dans un format qui leur est propre. Il faut donc faire appel à des traducteurs (que nous appellons ´drivers´) qui traduisent dans un format global les résultats transmis. Ces drivers doivent prendre en considération des éléments tels que le type de machine, leur logique de travail, etc... Le programme de vérification (alias Monsieur X), peut ainsi "comprendre" l´ensemble des résultats transmis et aussi les mémoriser dans une base de données

Ouverture du problème

* Des intermédiaires peuvent prendre place dans cette chaîne de transfert
* Les machines d'analyses transmettent des résultats pendant toute la journée. A l'opposé, le programme de vérification n'est actif que quelques minutes par jour. Il est donc necessaire d'ajouter un niveau de mémorisation.

Cinq solutions

Il existe cinq solutions technologiques aux communications entre machines d'analyses et l'ordinateur principal.
Ces cinq solutions ainsi que quelques compléments technologiques et les prix d'achats de ces solutions vous sont présentés (prix pour la mise ONLINE de 4 machines):

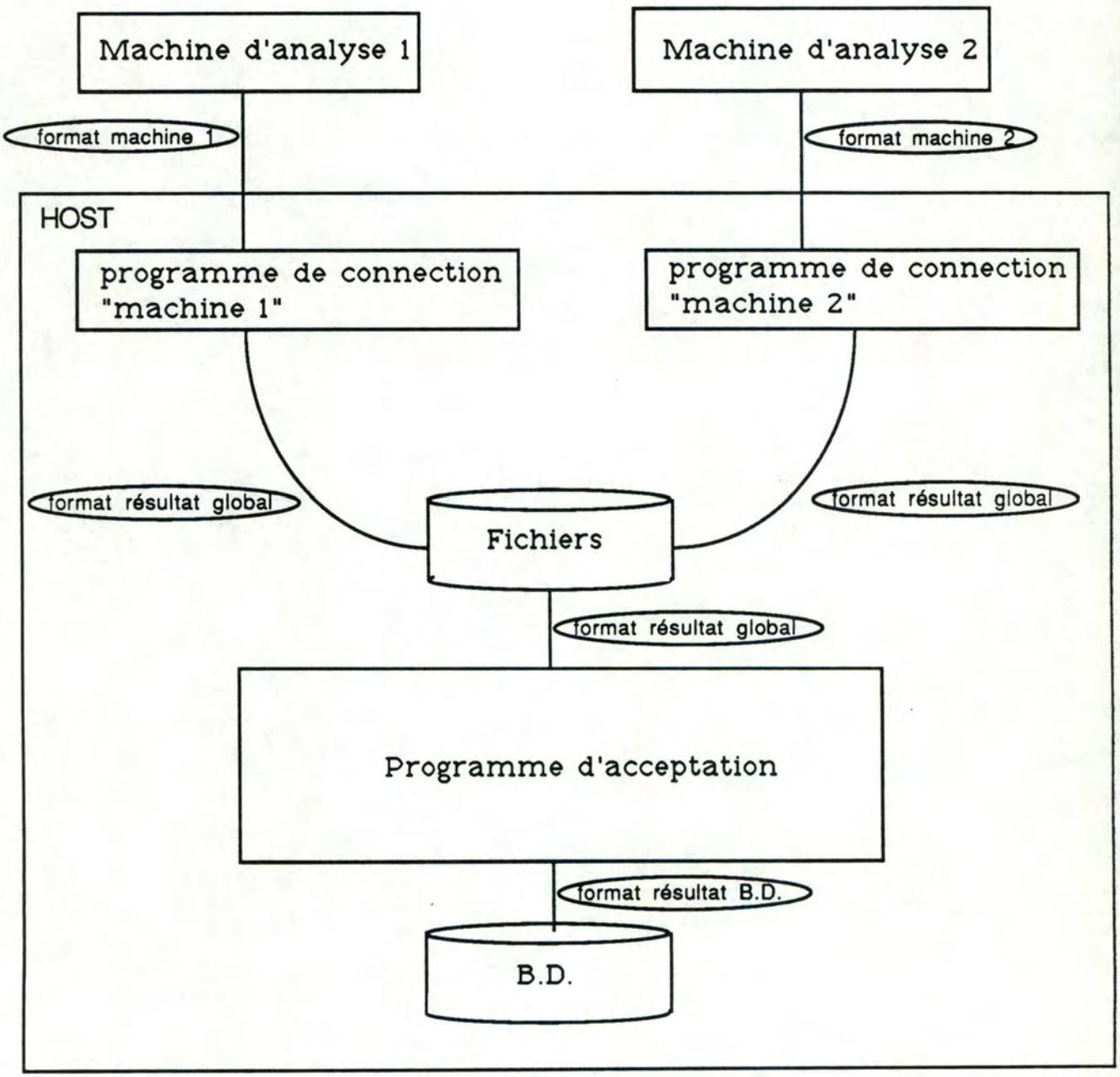
- connection sur le host
- micro-concentrateur
- système UNINA
- système à base de PC
- système PGP

Présentation des schémas de connection

Sur tous les schémas, seront repris:

- le type de connection
- le matériel nommé et représenté par un rectangle
- les programmes nommés et représentés par un rectangle
- les formats de données nommés et représentés par un ovale
- les connections représentées par un trait entre deux machines

- *connection des machines d'analyses directement sur l'ordinateur principal*



La connection directe n'est réalisable que par l'adjonction de programmes de connection à chaque machine d'analyses. Ces programmes sont spécifiques à la machine d'analyses et il sont bien sûr différents. Il me semble toutefois qu'il soit possible de trouver une base commune à tous ceux-ci et donc de réduire le coût de réalisation pour une machine supplémentaire.

La norme RS232C exige que la distance entre deux systèmes informatiques soit inférieur à 15 mètres.

Les investissements

- Option unidirectionnelle

  * coût du soft de communication: 50000 FB par machine
  * coût des cables : 35 FB / mètre

  Pour quatre machines :
  * 50000 * 4 machines = 200000 FB
  * 35 * 40 Mètres = 1400 FB

  Total : 201400 FB

- Option bidirectionnelle

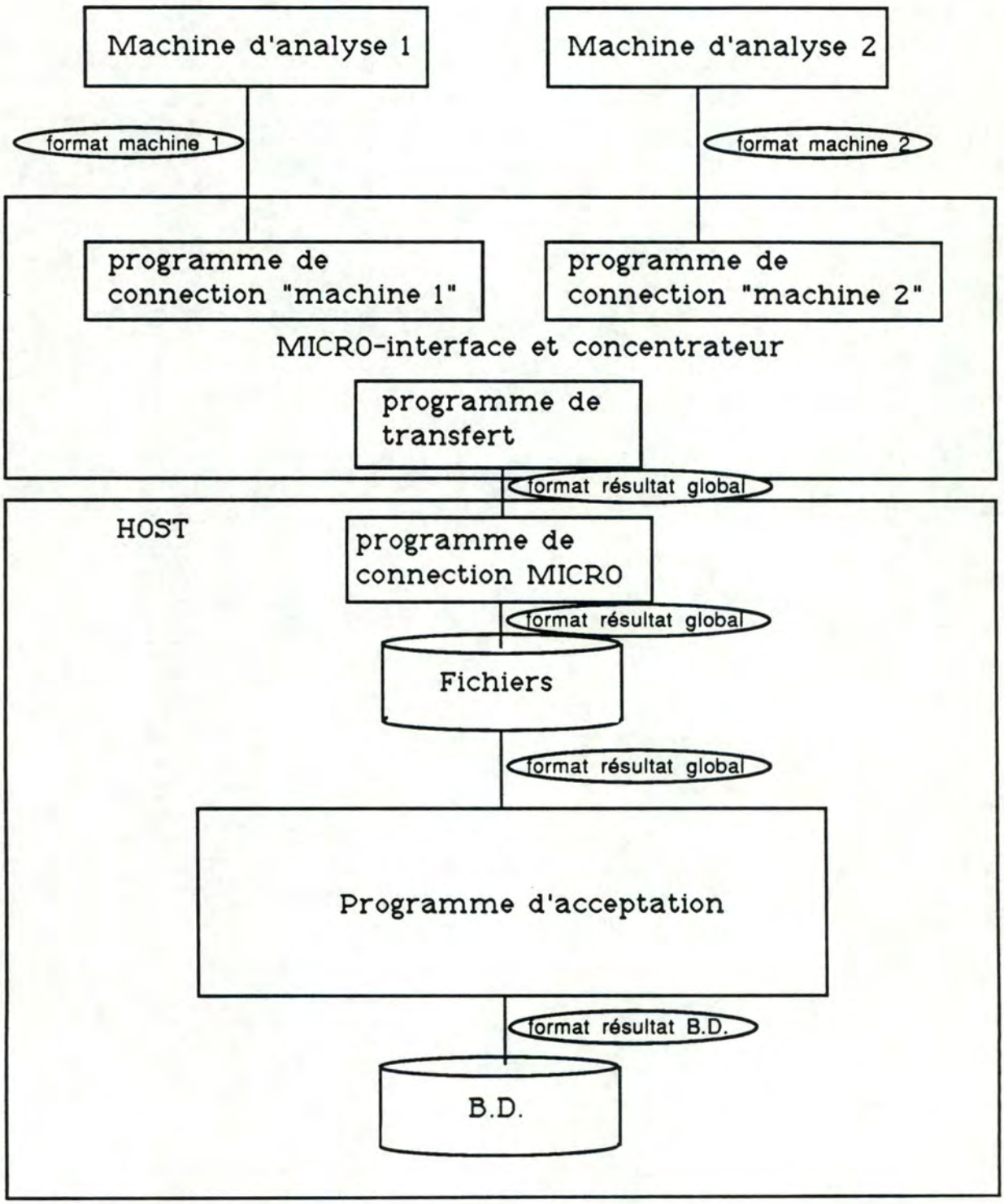
  Aucune modification hardware ne doit étre apportée au système de base. Seules quelques modifications seront apportées ainsi qu'un nouveau programme.

  * modification soft : 10000 FB par machine
  * nouveau programme : 50000 FB

  Pour quatre machines
  * 10000 * 4 machines = 40000 FB
  * nouveau soft = 50000

  Total : 90000 FB

- *connection des machines d'analyses sur un micro-ordinateur faisant office de concentrateur*



L'utilisation d'un PC AT (ou compatible) et d'une carte de communication (8 portes) peut servir de concentrateur.

Les investissements

- Option unidirectionnelle

  * PC AT + 8 portes de communication : 250000 FB
  * coût du soft pour chaque machine sur le PC : 50000
  * coût du soft pour la communication avec le host sur PC : 50000
  * coût du soft de communication avec le PC sur le host : 50000
  * Cout du soft de mise en forme des résultats en fonction des machines : 50000 FB par machine
  * coût des cables : 35 FB / Mètre

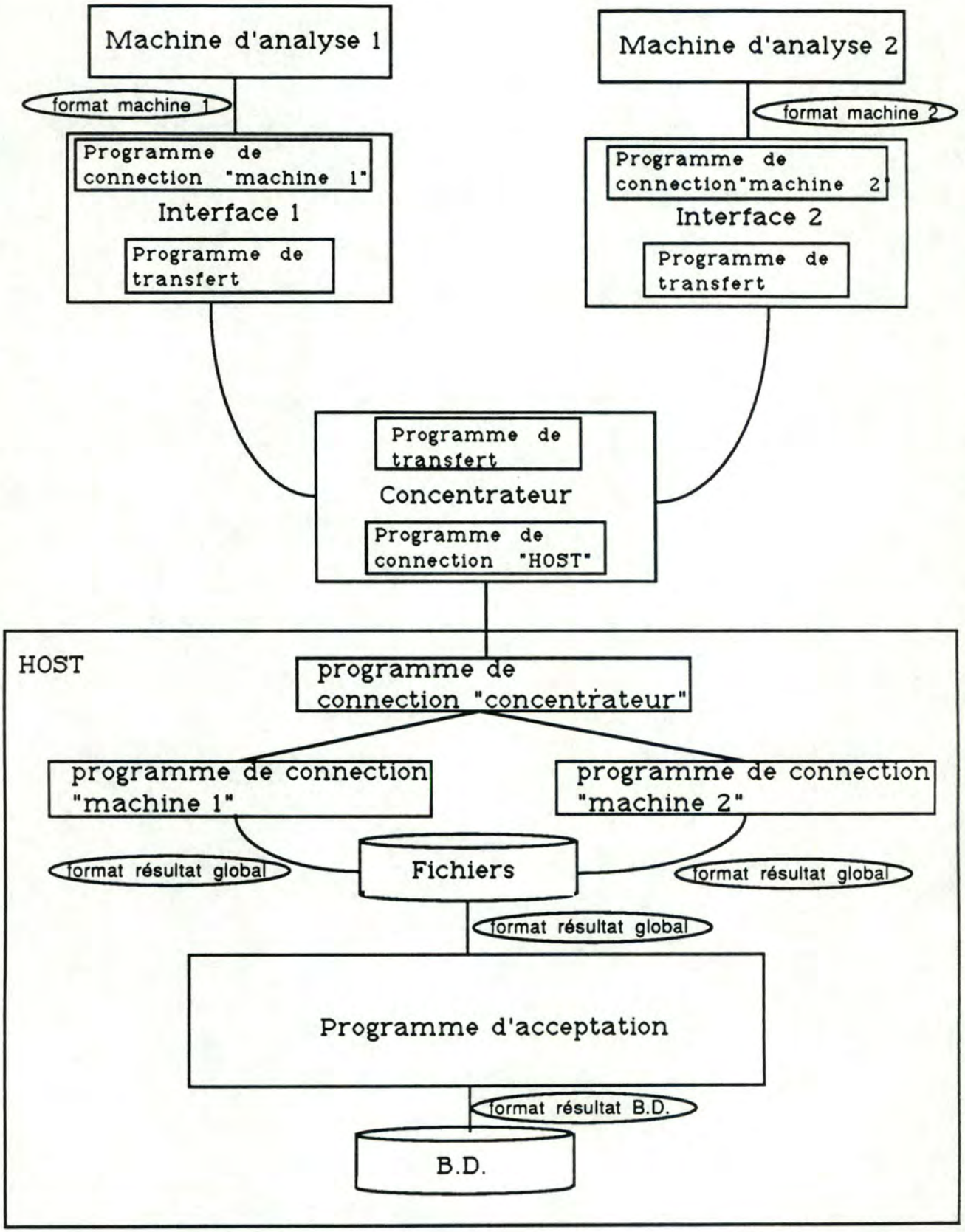
  Pour quatre machines
  * 250000 FB
  * 50000 * 4 machines = 200000 FB
  * 50000 FB
  * 50000 FB
  * 50000 * 4 machines = 200000 FB
  * 35 * 40 mètres = 1400 FB

  Total : 751400 FB

- Option bidirectionnelle

Cette option n'est pas étudiée vu les difficultés de réalisations

- *connection des machines d'analyses à l'ordinateur principal par l'adjonction d'interfaces et d'un concentrateur spécialisé de marque UNINA*



Ce système complexe permet une gestion des résultats en temps réel et une mémorisation de ceux-ci à trois niveaux.

Les investissements

Nous avons déjà effectué les calculs lors de la présentation du système UNINA. Vous pouvez vous y référer.

- Option unidirectionnelle

Les programmes réalisés sur le host se ventilent ainsi
* programme de connection avec le système UNINA
* programme de mise en forme en fonction des machines d'analyses

Pour quatres machines
* matériel UNINA : 485500 FB
* soft de connection : 50000 FB
* soft de mise en forme : 50000 * 4 = 200000
* coût des cables: 35 * 40 mètres = 1400

Total : 736400 FB

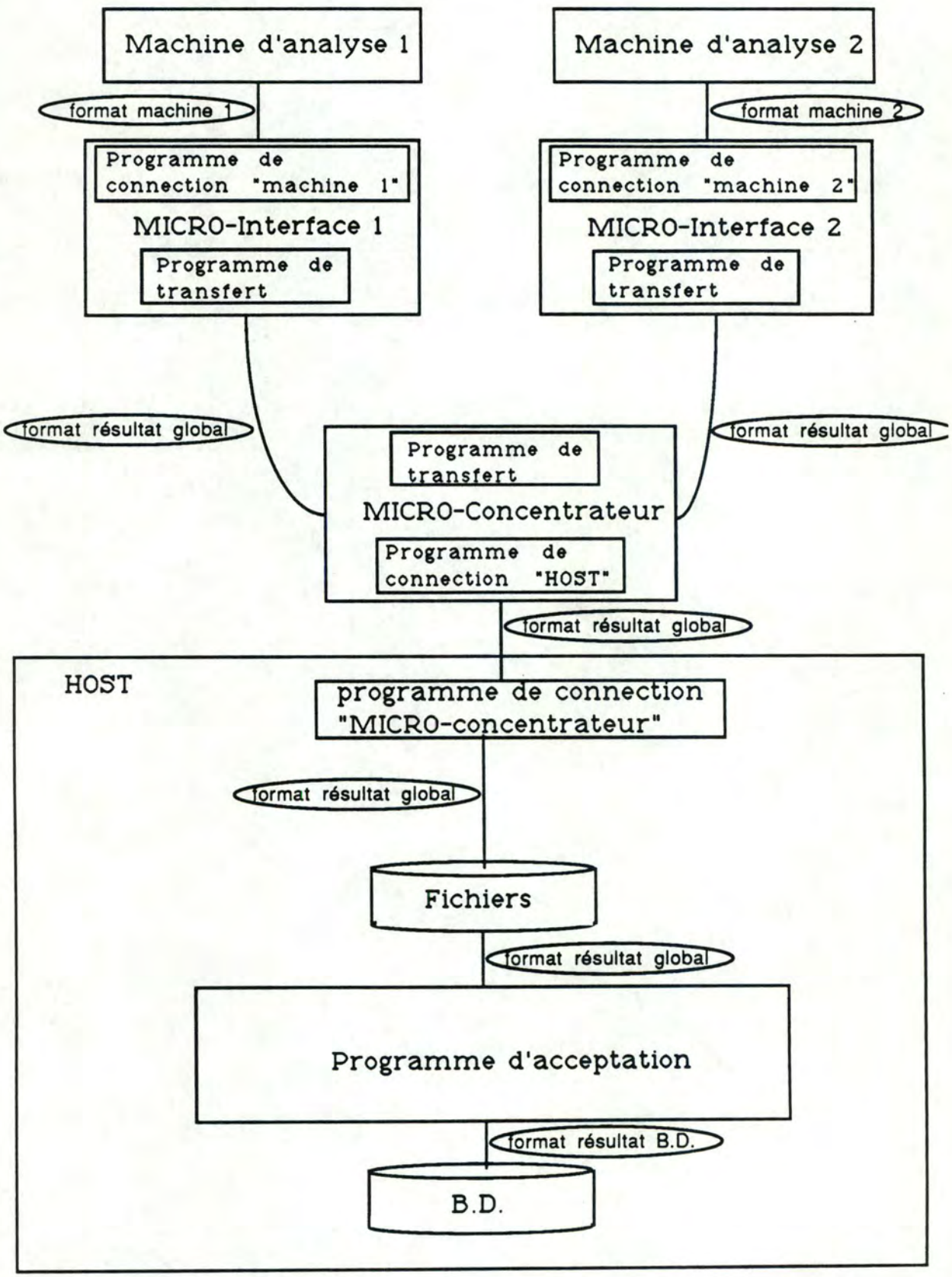
- Option bidirectionnelle

Le programme de connection devra être refait complètement et un nouveau programme ajouté pour chaque machine

Pour quatre machines
* MAJ système UNINA : 186000
* MAJ programme de connection : 50000
* programme BI 50000 * 4 = 200000

Total : 436000 FB de MAJ.

- *connection des machines d'analyses à des micro-ordinateurs eux-mêmes reliés à un autre micro utilisé comme concentrateur*



Ce système permet de ne plus travailler en temps réel ce qui facilite les softs de communication. Une nouvelle opération existe alors pour les laborantins : envoyer les résultats en fin de journée. Les interfaces étant programmables (PC) les résultats de toutes les machines auront le même format et un programme de connection global est suffisant.

Les investissements

- Option unidirectionnelle

  * coût du concentrateur : PC compatible 30000 FB
  * coût d'une interface : pc compatible 30000 FB
  * soft de connection du PC sur la machine : 35000 FB
  * soft de communication entre PC : 35000 FB
  * soft de communication avec le host : 50000
  * soft de communication avec le concentrateur : 50000
  * prix des cables : 35 FB / mètre

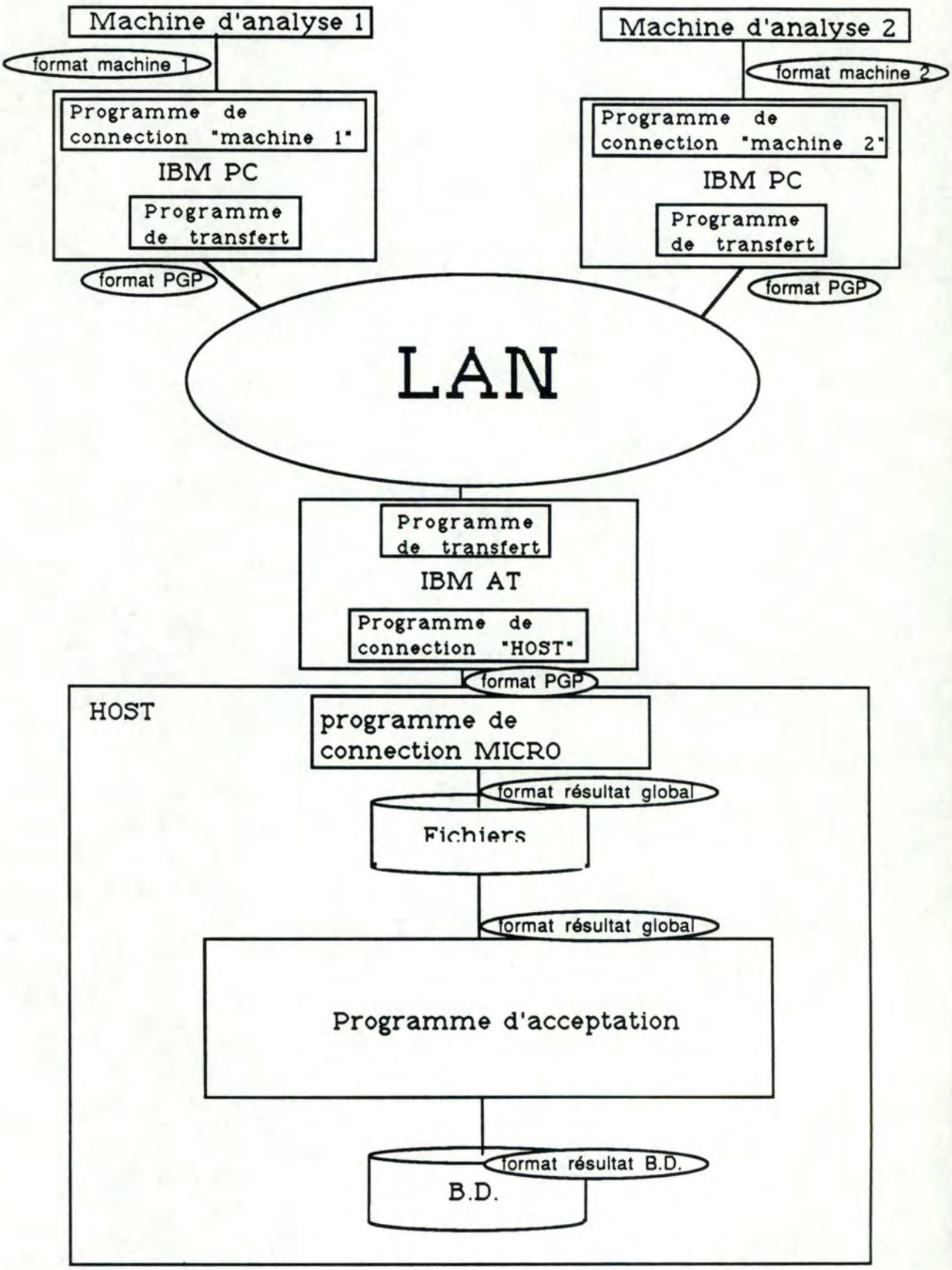
  Pour quatre machines
  * concentrateur : 30000 FB
  * interfaces : 30000 * 4 machines = 120000 FB
  * soft connection machines : 35000 * 4 = 140000 FB
  * soft de communication entre PC : 35000 FB
  * soft de connection avec le host : 50000 FB
  * soft de connection avec le pc : 50000 FB
  * cables : 35 * 40 mètres = 1400 FB

  Total : 496400

- Option bidirectionnelle

  Aucune modification hardware n'est necessaire
  Les modifications des programmes de connection ne semblent pas importantes. Les prix seront alors inchangés. Un nouveau programme sur le host devra être implémenté coût : 50000 FB

- *un système composé d'IBM pc est proposé en Belgique par la firme PGP. Cette solution consiste en un réseau d'IBM pc qui sont eux-même connectés aux machines d'analyses.*



Ce système travail en temps réel et permet une première vérification et acceptation au niveau des interfaces. Un format global est commuiqué au host, quelle que soit la machine d´analyses.

Les investissements

- Option unidirectionnelle

  * coût du concentrateur
  * coût des interfaces
  * coût du soft de connection avec les machines d´analyses
  * coût du soft de connection avec le concentrateur sur le Host
  * les coûts des cables, de la mise en place etc... sont à la charge de la firme PGP

  Pour quatre machines
  * concentrateur : 250000 FB
  * interface : 75000 * 4 machines = 300000 FB
  * soft interface : 50000 * 4 machines = 200000 FB
  * soft connection concentrateur : 50000 FB

  Total : 800000 FB

- Option bidirectionnelle

  * le soft de connection des machines doit être modifié
  * un nouveau programme sur le host doit être apporté

  Pour quatre machines
  * soft interface : 25000 * 4 machines = 100000 FB
  * nouveau soft : 50000 FB

  Total des MAJ : 150000 FB

Choix d'une solution

Calcul de rentabilité

Dans ce calcul, nous allons utiliser les modèles classiques du calcul économique basés sur le concept de valeur actualisée. Ces modèles reposent sur l'estimation de trois échéanciers.

- L'échéancier des investissements Ij, (1<j<n) n indiquant la durée de vie estimée du projet.
- L'échéancier des dépenses de fonctionnement Dj (1<j<n).
- L'échéancier des économies de fonctionnement Rj (1<j<n).

A partir de ces trois échéanciers, on peut calculer le bénéfice actualisé:

$$BA = \sum_{i}^{n} \frac{R_i - D_i - I_i}{1 + i}$$

Nous allons classer ces solutions par ordre croissant de rentabilité.

La durée de vie de cet investissement est de 5 ans.

Nous pouvons maintenant passer aux calculs.

En commun aux cinq systèmes, nous avons:
- Une dépense de fonctionnement importante consiste en l'utilisation du programme de vérification des données.
- Les économies de fonctionnement sont principalement dues au non encodage des résultats de ces machines d'analyses.
- Une dépense d'investissement : la réalisation et mise au point du programme de vérification des données.

Les économies et dépenses de fonctionnement étant communes à tous les systèmes, la seule dépense d'investissement fait la différence. C'est pourquoi je ne vous propose qu'un synoptique des investisements.

Dans la première proposition une autre dépense de fonctionnement consiste en l'utilisation exclusive d'une porte de communication sur le host pour chaque machine d'analyses connectée. Dans le cadre du laboratoire, cette utilisation exclusive aurait posé problème. En effet, actuellement quasiment toutes les portes sont utilisées par des terminaux interactifs. Il est toujours possible de faire l'investissement de huit nouvelles portes sur le host. Coût : plus ou moins 150000 FB.

Synoptique des investissements

La première colonne reprend le type de connection.
La seconde colonne donne le prix de la connection unidirectionnelle de 4 machines.
La troisième colonne donne le prix de la connection bidirectionnelles de 4 machines.
La quatrième colonne donne le prix d'une connection supplémentaire.
La cinquième colonne donne l'équivalent économique en minutes et par jour de l'utilisation d'une secrétaire
La sixième colonne donne l'équivalent économique en minutes par jour et par machine de l'utilisation d'une secrétaire

	UNI	BI	marginal	% secrétaire	%machines
sur le host	351400	440000	50000	35 minutes	9 minutes
micro-concentrateur	526400	570000	85000	50 minutes	13 minutes
système UNINA	736400	1272400	100000	70 minutes	18 minutes
système PGP	800000	950000	125000	80 minutes	20 minutes

A la lecture de ces chiffres, le système de connection directe sur le host est de loin le moins cher, malgré l'achat d'un concentrateur huit portes. Cet achat n'est pas toujours necessaire.

Actuellement, pour être rentable (par rapport à l'engagement d'une secrétaire) il serait necessaire que ce système de ONLINE par système UNINA permette un gain d'un peu plus d'une heure par jour. Ceci est totalement impossible ! Si par contre, huit machines étaient mise ONLINE, il faudrait faire un gain d'un peu plus de 1h30 par jour, ce qui fait un peu plus de 12 minutes par machine. Pour de grosses journées, le système devient rentable.

Certains bénéfices ne sont pas quantifiables.

Rappelez-vous que tout le programme de gestion du laboratoire existe sur le marché et l´option de ON-LINE est probablement importante. De plus le système UNINA est certainement un des plus répandu dans les laboratoires.

La décision est un acte volontariste.

En fait toute cette analyse ne servait à rien ! En effet l´achat du système UNINA ayant déjà été effectué, il était certain que la décision serait de reprendre ce système. Tout ce qui m´est permis dans cette analyse est de critiquer ce choix, non pas pour le laboratoire Janssens et Dubois mais dans le cas (peu probable) d´une autre informatisation de laboratoire.

Analyse conceptuelle

Restrictions

La décision de travailler avec le système UNINA est prise.

Cette analyse se bornera aux problèmes du ON-LINE et non pas à la globalité du problème.

Le scénario proposé dans la description du diagramme de flux est celui qui a été la base de l'analyse. Il faut bien comprendre que les scénarios sont presque aussi nombreux que ce que l'imagination vous propose. Jusqu'a présent cette analyse globale n'a jamais été prise en défaut mais je redoute le jour ou l'on me parlera d'un scénario digne d'un Godart en difficultés et d'un Pialat en rebondissements. Ce jour là, il me sera loisible de recommencer ...

Décomposition des traitements

Projet :

informatisation des traitements courants d'un laboratoire.

Application :

réalisation de la connection hardware et software entre des machines d'analyses médicales et l'ordinateur principal ainsi qu'un programme qui permette de s'assurer du transfert correct des résultats par cette voie.

Identification :

* interactions faibles et de facon ponctuelle (une fois par acceptation):
  - les JBL
  - les résultats d'analyses
* flux homogène d'information
  résultats des machines ON-LINE

Phases et description succincte de celles-ci :

* Numérotation des demandes d'analyses et des échantillons: les demandes et leurs échantillons respectifs reçoivent un numéro attribué dans l'ordre croissant. Ce numéro est constitué de 6 chiffres et devient le seul moyen de faire correspondre une demande et un échantillon.
* Encodage des analyses : l'ensemble des analyses de toutes les demandes sont encodées ainsi que des renseignements "mutuelle".
* Sortie des JBL : les analyses encodées sont rassemblées par services et constituent ainsi les nouvelles JBL qui sont sorties sur imprimante.
* Programmation des analyses : programmation manuelle des analyses à effectuer et constitution des JBL.

* Préparation des échantillons : certains échantillons doivent subir des traitements ou être transférés dans des récipients spécifiques à certaines machines.
* Programmation de la machine d'analyses : cette programmation consiste à choisir les analyses à effectuér sur une des machines d'analyses. Ainsi une machine d'analyses n'effectue pas toutes les analyses pour tous les échantillons.
* Analyse : acte biologique pour permettre la mesure quantitative et/ou l'identification qualitative de substances chimiques.
* Sortie des résultats sur imprimante : les résultats des machines d'analyses perfectionnées sont donnés par imprimante ou écran incorporé
* Recopiage des résultats sur JBL : l'ensemble des résultats sont notés sur les JBL.
* Encodage des résultats : le secrétariat encode les résultats des analyses effectuées par le laboratoire.

Les phases qui suivent sont expliquées en détails. En effet, seules ces phases ne sont pas décrites dans l'analyse globale du laboratoire.

Transfert des résultats : cette phase doit mettre en oeuvre des protocoles de connection spécifiques avec vérification du transfert. En plus de ce transfert, un sauvetage est effectué dans un fichier à format unique.

Cette phase requiert l'utilisation d'une porte sur le host pendant toute la durée du transfert (toute la journée)

Informations en entrée
- identification de la machine
- identification de l'analyse
- numéro de séquence sur cette machine
- valeur du résultat
- mode de travail de la machine

Information en sortie
- fichier de résulats à format unique

Vérification : cette phase sera utilisée pour vérifier:

- que les transmissions de données se sont correctement déroulées
- que toutes les analyses demandées ont bien été effectuées
- que toutes les analyses effectuées ont bien été demandées.

Si lors de la vérification on remarque une erreur, il sera possible d'y remédier (correction, suppression, ajout, ...).

Cette phase requiert une personne et un terminal.

Informations en entrée

fichier des résultats transmis par la phase de transfert de résultats

fichier des JBL donné par la phase de programmation des analyses

identification de la machine vérifiée

Informations en sortie

transfert des résultats acceptés dans la base de données

Programmation automatique des machines : système automatique de programmation des machines d'analyses bi-directionnelle en fonction des JBL.

Cette phase requiert l'utilisation d'une porte de transmission sur le host durant la période de programmation des machines (quelques minutes par jour)

Informations en entrée

- identification de la machine
- identification de l'analyse
- fichier des JBL
- numéro de séquence
- mode de travail

Information en sortie

- suite de caractères nécessaire à la programmation de la machine d'analyses en fonction d'éléments tels que numéro de séquence, mode de travail, etc...

Fonction :
Nous négligerons cette étape trop laborieuse excepté pour les phases de programmation automatique des machines d'analyses, transfert des résultats et vérification.

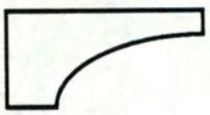
programmation automatique des machines :
- initialisation
- lecture dans la B.D.
- transfert vers la machines d'analyses
- choix de la machine

transfert des résultats :
- initialisation
- sauvetage dans fichier
- réception de données des machines d'analyses
- modification de format en fonction des machines d'analyses

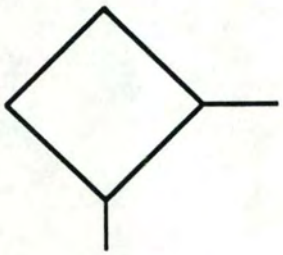
vérification :
- initialisation
- sauvetage B.D.
- lecture fichier
- éditeur de tableau
- fonctions diverses d'aide à la vérification

Explication des schémas de
la dynamique des traitements

Relation de déclenchement

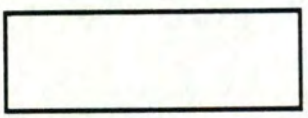


Message



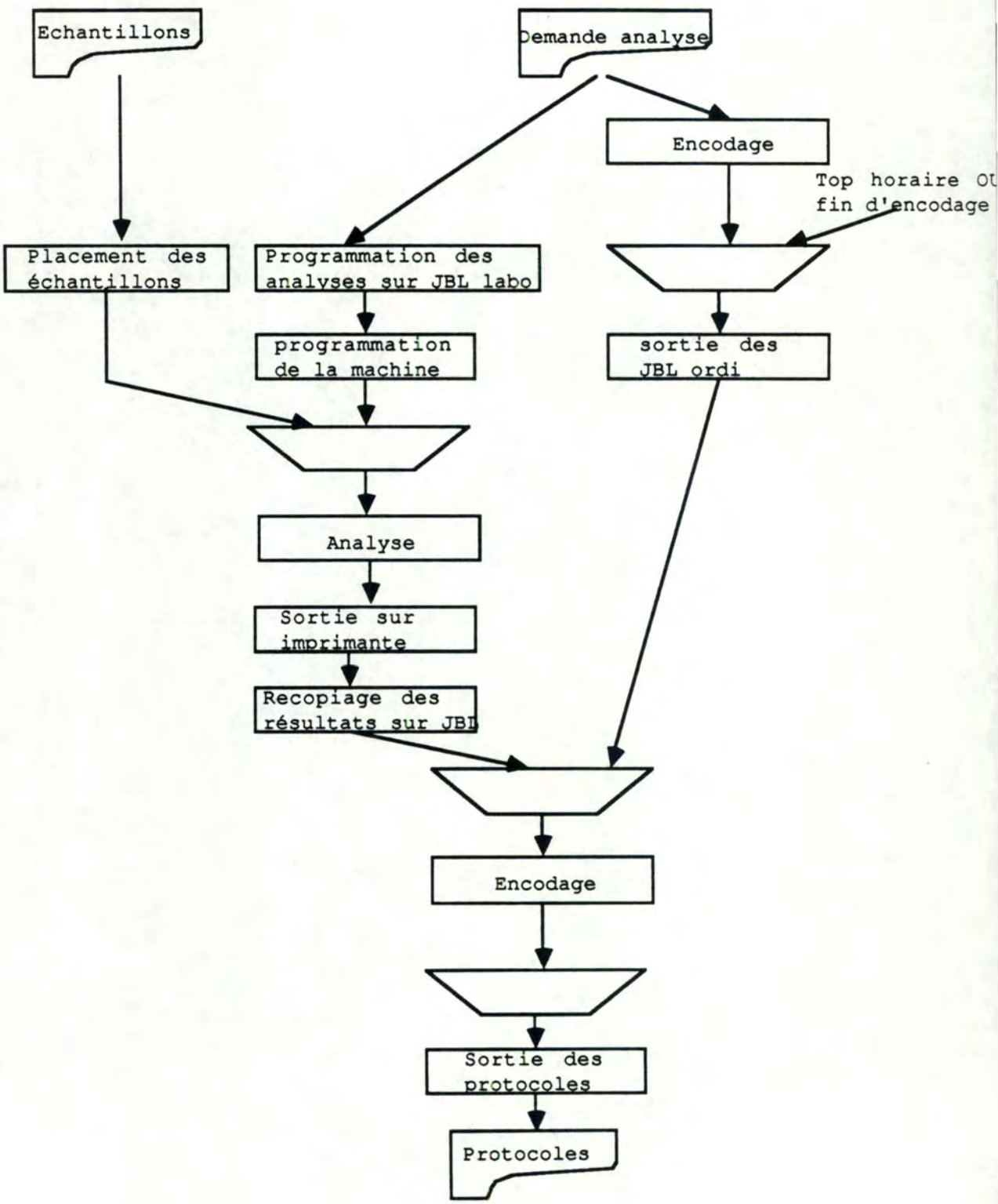
Enchaînement conditionnel

Enchaînement synchronisé

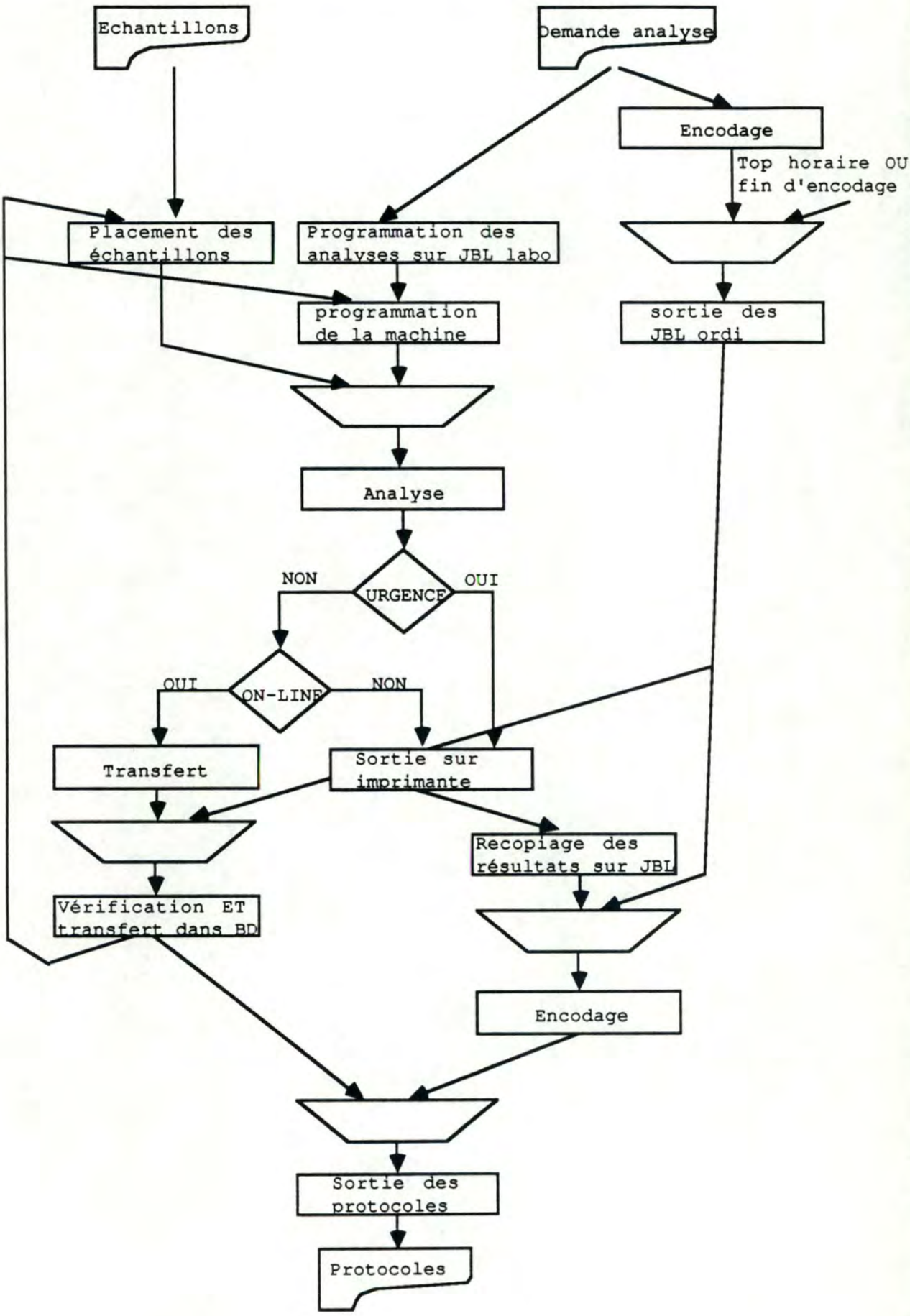


Processus (=phase)

Dynamique des traitements pour une solution sans transfert automatique des résultats



Dynamique des traitements pour une solution à base d'un ONLINE



Structuration des informations

Restrictions

Chaque phase devrait faire l'objet de cette étude. Je me limiterai pourtant au seules phases n'ayant pas encore fait l'objet de l'étude réalisée au laboratoire lors de l'informatisation: les phases liées au problème de l'ONLINE. Aucune intégration ne sera donc faite.

Phases

TRANSFERT:

Ensemble des entités :

- Machine d'analyses : machine d'analyses biologiques qui possède la possibilité de transmettre ses résultats par interface intégrée.

- String reçu : suite de caractères transmise par une machine d'analyses et possédant un système de checksum.

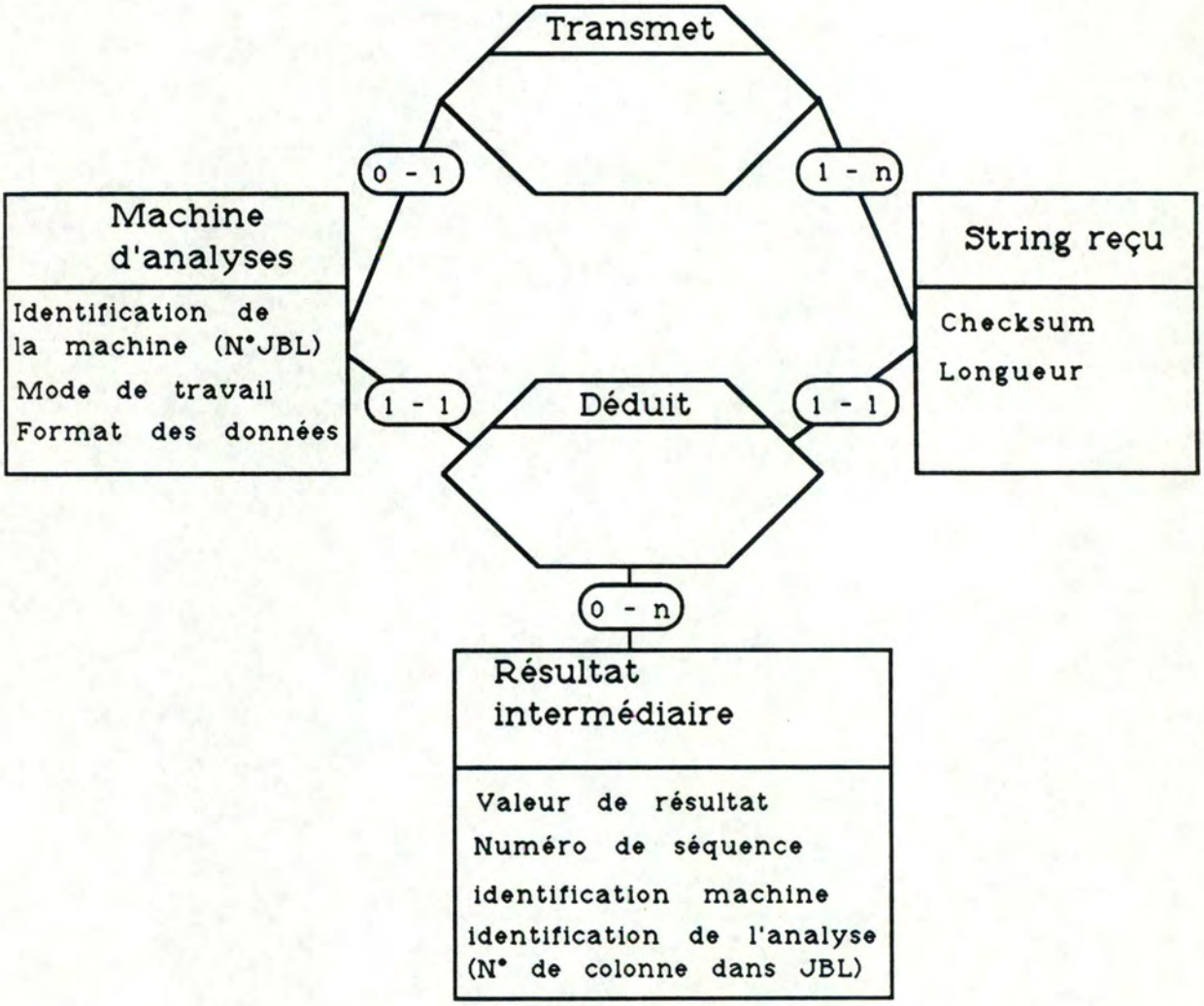
- Résultat intermédaire : résultat d'une analyse effectuée par une des machines d'analyses.

Ensemble des associations

- Transfert : exprime le lien qui existe entre la machine et le host: une suite de caractères.

- Déduit : exprime le lien qui existe entre les caractères transmis, la machine d'analyse et le résultat de l'analyse.

Représentation graphique


Transmet
0 - 1
1 - n
Machine d'analyses
Identification de la machine (N°JBL)
Mode de travail
Format des données
String reçu
Checksum
Longueur
1 - 1
Déduit
1 - 1
0 - n
Résultat intermédiaire
Valeur de résultat
Numéro de séquence
identification machine
identification de l'analyse (N° de colonne dans JBL)

ACCEPTATION :

Ensemble des entités :

Résultat intermédiaire : défini précédemment.

Demande d'analyses : ensemble des analyses demandées par un médecin pour un patient.

Résultat B.D. : résultat qui est sauvé dans la base de données

Valeurs de référence : valeurs données pour une analyse qui caractérise un être sain.

JBL labo : job liste manuelle effectuée par un laborantin et completée des résultats des analyses.

JBL ordi : job liste automatique déduite des demandes d'analyses encodées.

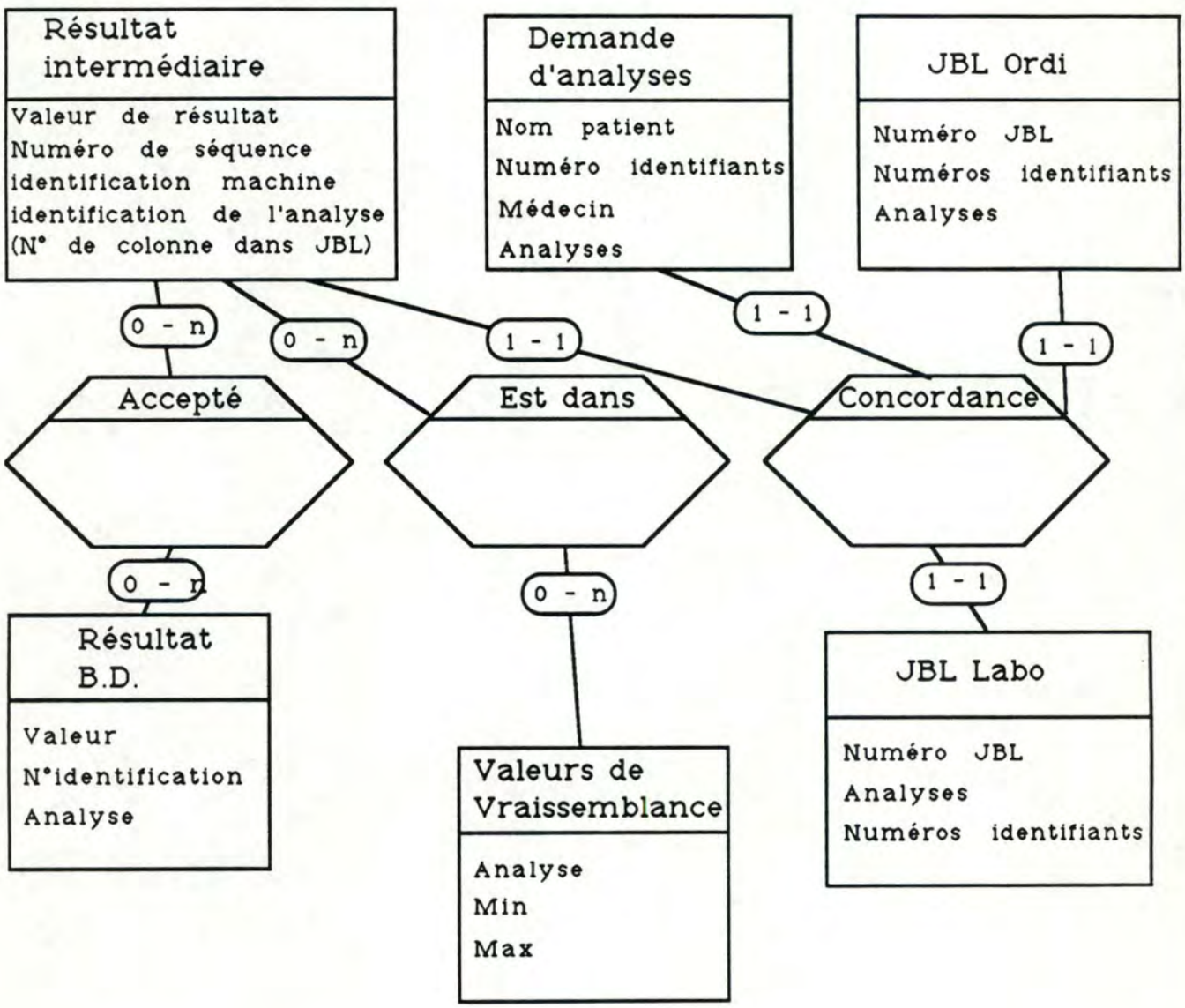
Ensemble des associations

- acceptation : exprime le transfert d'un résultat intermédiaire dans la base de donnée.

- concordance : exprime le lien qui existe entre le résultat transmis par la machine, le résultat noté sur la JBL labo, l'analyse demandée et l'analyse programmée.

- est dans : exprime le lien entre un résultat et ses valeurs de vraisemblance.

Représentation graphique


Résultat intermédiaire
Valeur de résultat
Numéro de séquence
identification machine
identification de l'analyse
(N° de colonne dans JBL)
Demande d'analyses
Nom patient
Numéro identifiants
Médecin
Analyses
JBL Ordi
Numéro JBL
Numéros identifiants
Analyses
0 - n
0 - n
1 - 1
1 - 1
1 - 1
Accepté
Est dans
Concordance
0 - n
0 - n
1 - 1
Résultat B.D.
Valeur
N°identification
Analyse
Valeurs de Vraissemblance
Analyse
Min
Max
JBL Labo
Numéro JBL
Analyses
Numéros identifiants

PROGRAMMATION AUTOMATIQUE DES MACHINES

Ensemble des entités

- Machine : défini précédement.

- String transmis : suite de caractères transmis pour programmer la machine d´analyses.

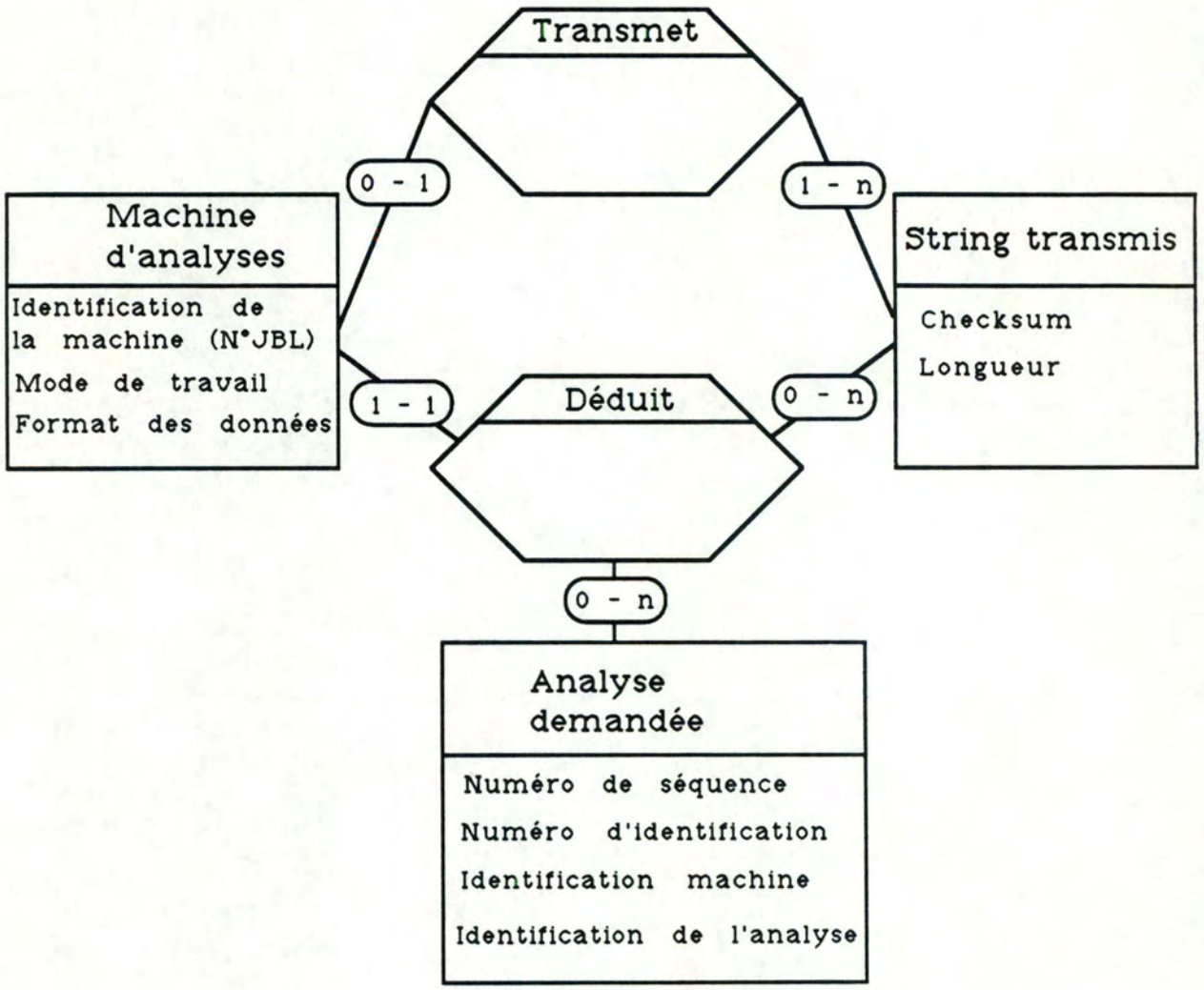
- Analyse demandée : analyse qui fait partie d´une demande d´analyses.

Ensemble des associations

- transfert : exprime le lien qui existe entre la machine et le host: une suite de caractères.

- déduit : exprime le lien qui existe entre les caractères transmis, la machine d´analyses et l´analyse demandée.

Représentation graphique


Transmet
0 - 1
1 - n
Machine d'analyses
Identification de la machine (N°JBL)
Mode de travail
Format des données
String transmis
Checksum
Longueur
1 - 1
Déduit
0 - n
0 - n
Analyse demandée
Numéro de séquence
Numéro d'identification
Identification machine
Identification de l'analyse

Nous nous avons définis tout ce qui était nécessaire à la simplification de l'étape qui suit: le développement de logiciel.

CHAPITRE III

METHODOLOGIE DE DEVELOPPEMENT DE LOGICIEL

METHODOLOGIE DE DEVELOPPEMENT

Explications préalables

Tous les concepts et méthodes proposés ci-après sont repris du cours donné par Monsieur Van Lamsweerde "Methodologie de développement de logiciels".

Nous allons essayer de donner les étapes qui ont suivi l'analyse fonctionnelle dans le développement des logiciels. Nous spécifierons ces programmes puis nous chercherons à développer une architecture logicielle et enfin nous prouverons la validation du logiciel par les tests appropriés. Tout ceci sera accompagné d'une documentation complète.

Spécifications fonctionnelles.

Nous utiliserons les spécifications par règles, c'est à dire un ensemble non structuré de phrases descriptives. L'utilisation de cette méthode comporte quelques dangers

- Le bruit : élément de texte qui n'apporte aucune information.
- Les silences : Caractéristiques du problème auquel ne correspond aucun élément du texte.
- Les contradictions : éléments de texte qui définissent de manière incompatible une même caractéristique d'un probème.
- L'ambiguité : éléments de texte qui permettent de comprendre une même caractéristique du problème d'au moins deux manières.
- Les références en avant : élément de texte qui fait usage de caractéristiques des problèmes définis plus loin dans le texte.
- Les repentirs : élément de texte qui définit une caractéristique du problème de manière trop tardive.

Développement d'une architecture logicielle.

Etablir une architecture logicielle c'est structurer le système à développer en un ensemble de composants qui

- jouissent de certaines qualités désirées
- comprennent les traitement et les structures de données qui réalisent les différentes fonctions et données identifiées au cours de l'analyse fonctionnelle.
- ont entre eux des relations bien définies
- obéissent à des contraintes données

Dans le developpement d'un logiciel, certains points doivent être respectés au mieux. Il est conseillé (pour soi et les autres) de faire un logiciel dont :

la validité n'est pas à remettre en cause. Il s'agit d'être sûr que le logiciel fait correctement ce qu'on lui demande et pas plus.

la robustesse est élevée. Veiller à gérer toutes les erreurs pouvant venir de l'extérieur. Ce point est certainement le plus difficile à mettre en oeuvre car il faut penser à tous les cas de figure au niveau fichiers, système et humain.

la portabilité est élevée. Ceci pourrait servir dans d'autres contextes de programmation. Il faut toutefois se rendre compte que dans notre cas nous ne serons pas indépendants du langage, de l'OS et de ses utilitaires (RTL, STALET, ..), etc... Il sera indépendant des machines à mettre ON-LINE.

l'efficacité est la meilleure. Point crucial de ce logiciel ou la recherche du meilleur compromis vitesse / mémoire n'a pas encore été défini précisément.

la maintenabilité est facile. Ceci dépend de la manière de programmer et de la documentation proposée.

Principe de structuration et hiérarchisation

Structurer un programme consiste à l´organiser selon un système de niveaux distincts et ordonnés.
"On considère qu'une structure est hiérarchisée si et seulement s'il existe une relation (R) entre ses composants qui permet de définir des niveaux tels que le niveau 0 est l'ensemble des composants appelé 'A' tel qu'il n'existe aucun composant 'B' avec la relation R(A,B) et tel que le niveau i est l'ensemble des composants 'A' tel qu'il existe un ou plusieurs 'B' de niveau i-1 tel que R(A,B) et tel qu'on ait R(A,C) avec 'C' de niveau inférieur ou égal à i-1".

Nous prendrons, pour cette application, la relation ´utilise´. R(A,B) : A ´utilise´ B ou le fonctionnement correct de ´A´ dépend de la disponibilité d´une version correcte de ´B´.

Après la décomposition en niveaux, nous allons modulariser la structure c-a-d

- pour chaque niveau, identifier les différents composants
- spécifier ces composants
- définir les relations entre ceux-ci.

Il sera alors possible de le représenter sous forme d´un graphe dans lequel les noeuds correspondent à un module et les arcs à une relation.
Les qualités d´un module seront appréciées en fonction de

- sa capacité de cacher de l´information: tout module doit cacher un maximum de décisions, de désigns, de conceptions aux autres modules en relation avec lui.
- sa forte cohésion interne: le degré d´interdépendance logique entre conditions et actions doit être aussi faible que possible.
- son faible degré de couplage: le nombre et la complexité de connections statiques avec d´autre modules doit être limité.

La validation du logiciel.

Cette validation se fera par l'approche des tests. Ceux-ci servent à établir la présence d'erreur.

Principe: en fonction d'un critère de couverture on cherche à se définir un bon jeu de tests tel que ce critère soit satisfait.

Nous utiliserons deux critères de couverture

Black-box : le but est de couvrir chaque cas limite de propriété d'une entrée et l'ensemble des causes-effets de la spécification.

Intégration : on cherche à couvrir toute combinaison possible d'appels de modules. J'utiliserai la méthode incrémentale qui consiste à tester un module en combinaison avec tous les autres modules déja testés.

P R O G R A M M E D E V A L I D A T I O N

Spécification du programme de validation

Le programme dont nous allons définir les spécifications fonctionnelles, est le complémentaire indispensable du programme de transmission de données par ON-LINE. Ce programme sera utilisé pour vérifier :

- que les transmissions de données se sont correctement déroulées
- que toutes les analyses demandées ont bien été effectuées
- que toutes les analyses effectuées ont bien été demandées.

Si lors d'une vérification on remarque une erreur, il sera possible d'y remédier grâce à un éditeur qui fera partie du programme.

Il nous a été demandé de rendre ce programme aussi général que possible. il faut toutefois limiter cette généralité:

le programme ne sera pas indépendant

- du type du système d'exploitation utilisé (VAX-VMS)
- du type de représentation des données (cfr programme actuel)

Toutefois, nous le rendrons le plus indépendant possible

- du type de représentation des données transmises par les automates d'analyse
- du type de connection nécessaire pour ces automates

Rem:ces deux conditions permettraient une connection plus aisée pour la plupart des automates d'analyses médicales existants.

Utilisation d´un éditeur

a) Titre général
Titre commun à toutes les fonctions du programme ´LABO´.

b) Titre du programme
Ce titre est le nom de la fonction dans ce programme ´LABO´. Il est composé de deux choses:
- du terme acceptation indiquant qu´on se trouve bien dans le programme d´acceptation du ONLINE.
- du titre de la JBL concernée par cette acceptation. Actuellement ce programme ne peut être utilisé que pour deux JBL différentes : Hitachi et Coulter (d´autres possibilités sont à l´étude : Cobas, BNA).

c) La ligne suivante est composée de signes dont voici la signification
´god´ : rapelle que la colonne est réservée aux numéro de godet utilisé pour ces analyses.
´ident´ : donne la colonne des numéros identifiants des demandes d´analyses. (000000 - 999999)
les noms d´analyses : cinq lettres maximum donnent le code de l´analyse. Ces codes sont définis par ailleurs (sigana.dat).

d) Le tableau
Les résultats des machines d´analyses sont dans la matrice ayant pour abscisses les noms des analyses et pour ordonnées les numéros d´identifiants ainsi que les numéros de godets associés. Les résultats sont de quatre types différents
- anormaux : les résultats sont en-dehors des normes de vraisemblance consacrées à cette analyse pour des sujets sains. (Ils apparaissent en gras à l´écran)
- superflus : les résultats sont transmis par la machine bien qu´ils n´aient pas été programmés par le secrétariat. Ces résultats peuvent ne correspondre à aucune demande. (Ils apparaissent en clignotant à l´écran)
- manquants : ce sont les résultats programmés par le secrétariat mais qui n´ont pas été transmis par le ONLINE.

(ceux-ci sont représentés par des barres horizontales)
- non-douteux : ce sont les résultats qui semblent être corrects (ni anormaux, ni superflus, ni manquants).

e) Le curseur
Ce curseur est représenté par un '>' clignotant. Ce curseur donne la position courante, c'est-à-dire donne le résultat, la ligne ou la colonne sur lesquels des modifications peuvent être effectuées (résultat,ligne et colonne courante).

f) La ligne de commandes
Cette ligne reprend le nom de la fonction dans laquelle on se trouve ainsi que toutes les possibilités liées à celle-ci. Si les possibilités ne tiennent pas en une seule ligne, il est possible de voir les autres en poussant sur <tab>. Si le fichier de commandes est inaccessible, la ligne est remplacée par 'no comment'

g) Ligne d'erreur et d'avertissement
Celle-ci est la dernière de l'écran et reprend toutes les erreurs et possibilités d'avertissement. Si le fichier des erreurs n'est pas accessible, la ligne 'undefined error' est présentée en cas d'erreur.

Première possibilité : le déplacement

Le déplacement curseur.

<les flèches> : celles-ci se trouvent sur la droite du clavier alphabétique et sur la gauche du bloc numérique. En poussant sur l'une de ces flèches, le curseur se déplace d'un résultat dans la direction de la flèche. Si vous vous trouvez sur un bord du tableau, et si vous cherchez à vous déplacer vers l'extérieur de celui-ci, un message d'erreur vous est présenté et le curseur ne quitte pas sa position d'origine. Si vous vous trouvez sur le bord d'une fenêtre sans être sur le bord du tableau, un déplacement curseur donne lieu à la présentation d'une nouvelle fenêtre et à un positionnement sur l'élément demandé (flèches).

Le déplacement des fenêtres

<ins><sel><pup><pdo> : les touches se trouvent juste au-dessus des flèches et dans les mêmes positions relatives. En poussant sur l'une de ces touches, la fenêtre se déplace dans la direction demandée. Si vous vous trouvez sur un des bords du tableau et que vous cherchez à vous déplacer vers l'extérieur de celui-ci, un message d'erreur vous est présenté et la fenêtre ainsi que le curseur ne changent pas de place. Si par contre, le déplacement est possible, le curseur se trouve sur l'élément courant si celui-ci existe toujours dans la nouvelle fenêtre, sinon le curseur se trouve sur le premier résultat rencontré dans le sens du déplacement.

Les déplacements rapides

Ces déplacements rapides se font en poussant deux touches successivement.

Les déplacements rapides dans la fenêtre

<'><flèches> : le curseur se déplace sur le résultat extrémal de la fenêtre, dans la direction de la flèche poussée.

Les déplacements rapides dans le tableau.

<'><déplacement fenêtre> : le curseur se déplace sur le résultat extrémal du tableau, dans la direction demandée.

Deuxième possibilité : les fonctions

'?' : Cette fonction permet de vous déplacer rapidement sur un résultat du tableau en donnant le numéro de ligne et le numéro de colonne de celui-ci. Si vous ne donnez aucun numéro de ligne (colonne) ou si vous introduisez une suite de caractères ne représentant pas un nombre, la ligne (colonne) courante est prise par défaut. Si vous donnez un nombre en dehors des possibilités du tableau, la position extrémale est prise par défaut. Si vous introduisez des caractères numériques puis d'autres , seul les caractères numériques sont pris en compte. S'il est nécessaire de déplacer la fenêtre, le calcul de celle-ci se fera en essayant de positionner le résultat recherché au centre de l'écran.

'N' : Cette fonction vous permet de vous déplacer rapidement sur le résultat de la première colonne associé à un numéro d'identifiant que vous avez donné. Si le numéro d'identifiant n'existe pas ou des caractères autres que des caractères numériques ont été introduits, un message d'erreur est proposé et aucun déplacement n'est effectué.

'F' : Recherche d'un résultat particulier. Cette recherche se fait sur les résultats qui se trouvent de la position courante à la fin du tableau en sachant qu'un parcours ligne par ligne est effectué. Si le résultat que vous introduisez n'existe pas ou n'est pas conforme au format d'un réel, un message d'erreur vous est proposé et aucun déplacement n'est effectué. Remarque importante. La recherche se fait indifféremment sur un réel, un entier ou un fractionnaire. MAIS si vous recherchez le résultat 1/8 vous devez introduire 8 comme élément recherché.

'R' : Vous pouvez supprimer toute une ligne de résultats en pressant la touche '-'. C'est la ligne courante qui est supprimée. Toutes les lignes suivantes sont alors remontées d'une ligne et la dernière ligne est sans résultat. Les résultats ainsi supprimés ne sont pas définitivement perdus,

ils se trouvent en fait dans un buffer. Vous pourrez donc récupérer le contenu grâce à une autre fonction qui sera examinée plus tard. Vous pourrez également ajouter une ligne de résultats en poussant sur '+'. Une ligne est ajoutée, précédant la ligne courante. Cette nouvelle ligne est bien sûr sans aucun résultat, les lignes de résultats suivantes sont descendues d'une ligne et la dernière ligne est mémorisée dans le buffer. Si vous pressez toute autre touche, aucune modification n'est effectuée.

'B' : Par cette touche vous allez substituer les résultats contenus dans le buffer, aux résultats de la ligne courante. Les résultats de la ligne courante sont définitivement perdus.

'I' : Cette fonction d'introduction automatique des résultats manquants va vous permettre de compléter les résultats du tableau en vous déplacant de résultat manquant en résultat manquant et en attendant l'introduction de celui-ci. L'introduction des résultats se fait par l'utilisation d'une fonction spécialisée supposée connue par l'utilisateur. Une modification toutefois: si vous tapez un point ('.') en première position, la fonction d'introduction automatique est stoppée et un message d'avertissement est affiché. Le déplacement se fait ligne par ligne et ce jusqu'à la fin du tableau ou l'arrêt volontaire de l'utilisateur.

'V' : Cette fonction vous permet de vous déplacer de résultat douteux en résultat douteux depuis le premier résultat jusqu'au dernier ou l'arrêt volontaire de l'utilisateur.
Voici les posssibilités qui vous sont offertes lorsque la fonction se trouve sur un résultat.
<space> : ceci permet de passer au résultat douteux suivant sans aucune modification de celui-ci.
<PF1> : arrêt de l'utilisation de cette fonction.
<return> : force un résultat douteux à devenir normal (un résultat anormal est considéré comme accepté, un résultat manquant est supprimé -la base de données est modifiée dans

ce sens- , un résultat superflu est accepté -la base de données est modifiée dans ce sens-
<back slash> : ceci permet de supprimer un résultat. (utilisé spécialement pour les résultats superflus)
<autre> : introduction d'un résultat. Remarque: si le résultat introduit est anormal, il ne sera pas corrigé. Une autre manière de stopper cette fonction est d'introduire un point ('.') en première position.

'Q','E' : Vous pouvez quitter le programme en enregistrant les modifications que vous venez d'apporter ('E') ou en ne les sauvant pas ('Q'). Lors du sauvetage, il est possible que des problèmes de concurrence entre fichiers apparaissent. Dans ce cas le programme cherche à le résoudre et s'il n'y arrive pas, il demande à l'utilisateur s'il doit essayer à nouveau ou non.

Nous allons reprendre toutes ces possibilités en les formalisant davantage.

DEPLACEMENT

Objectifs : Déplacer le résultat et/ou la page de résultats courante.

Messages d´entrée :
- position initiale du curseur
- élément, résultat courant
- page courante
- type de déplacement

Contraintes :
- déplacement du curseur/page dans la direction demandée si possible (Si on ne se trouve pas au bord du tableau).

Messages de sortie :
- nouvelle position du curseur et/ou nouvelle page courante
- le résultat courant est celui pointé par cette nouvelle position du curseur
- "Il n´existe pas de lignes/colonnes suivantes/précédente"

MODIFIER-RES

Objectifs : éditeur de résultats (modification, ajout, suppression d'un résultat)

Messages d'entrée :

résultat courant
position du résultat à l'écran
format du résultat
limite de vraisemblance pour cette analyse

Contraintes :

le nouveau résultat doit respecter exactement le format du résultat.

Messages de sortie :

la valeur du résultat courant est modifiée et affichée selon la représentation voulue

RECHERCHE-VALEUR

Objectifs : recherche d'une valeur précise dans tous les résultats qui suivent la position courante

Messages d'entrée :

- position courante
- l'ensemble des résultats
- la valeur recherchée

Contraintes :

- si la valeur recherchée n'existe pas alors il n'y a pas de modification de l'élément courant et un message d'avertissement est affiché.
- si la valeur recherchée ne correspond pas à un format connu, alors un message d'erreur est affiché et aucune modification n'est effectuée.
- si la valeur existe dans les résultats, on cherche à placer ce résultat au centre de l'écran et le résultat courant devient celui-ci

Messages de sortie :

- nouvelle position du curseur, nouveau résultat courant et page courante
- "Ce résultat n'existe pas"
- "Format incorrect"

INTRO-AUTOMATIQUE

Objectifs : introduire les résultats manquants de la position courante jusqu´au dernier résultat du tableau

Messages d´entrée :
  position courante
  l´ensemble des résultats manquants

Contraintes :
  l´introduction se fait ligne par ligne et en respectant les mêmes remarques pour modifier-res

Messages de sortie :
  position courante devient la position du dernier résultat

CORRECTION-AUTOMATIQUE

Objectifs : passer en revue, corriger ou confirmer les résultats anormaux

Messages d´entrée :
  position courante
  l´ensemble des résultat anormaux

Contraintes :
  la vérification se fait ligne par ligne avec contrainte de format (cfr modifier-res)

Messages de sortie :
  position courante du dernier résultat du tableau

AJOUT-DEL-LIGNE DE RES

Objectifs : ajouter ou supprimer une ligne de résultats

Messages d'entrée :
  ligne courante
  ajout ou suppression

Contraintes :
  pour un ajout, on décale l'ensemble des lignes d'une position vers le bas à partir de la ligne courante.
  pour une suppression, on décale l'ensemble des lignes d'une position vers le haut à partir de la ligne courante.

Messages de sortie :
  pour un ajout, la dernière ligne du tableau est mémorisée dans un buffer de ligne, la position/page courante est le nouveau résultat/page
  pour une suppression, la ligne supprimée est mémorisée dans un buffer de ligne, la position/page courante est le nouveau résultat/page

COPY-BUFFER

Objectifs : remplacement de la ligne courante par le contenu du buffer de ligne

Messages d'entrée :
  ligne courante
  buffer de ligne

Contraintes : /

Messages de sortie :
  nouvelle ligne courante

RECHERCHE-NO-REFERENCE

Objectifs : recherche d'une valeur de référence précise dans tous les numéros de référence du tableau

Messages d'entrée :
- valeur du numéro de référence
- l'ensemble des numéros de référence

Contraintes :
- si la valeur du numéro de référence n'existe pas alors il n'y a pas de modification de la ligne courante et un message d'avertissement est affiché
- si la valeur de numéro de référence ne respecte pas son format alors un message d'erreur est affiché

Messages de sortie :
- nouvelle ligne/page courante
- "Ce numéro de référence n'existe pas"
- "format incohérent"

SAUT

Objectifs : déplacement sur n'importe quel élément du "tableau"

Messages d'entrée :
- l'ensemble des lignes de résultats du tableau
- numéro de ligne
- numéro de colonne

Contraintes :
- les numéros de ligne/colonne doivent exister et respecter leur format respectif sinon aucune modification

Messages de sortie :
- nouveau résultat/page courant

CHOIX MACHINE-ANALYSE

Objectifs : choix de la machine d´analyses sur laquelle va s´effectuer la validation

Messages d´entrée :
l´ensemble des machines connectées

Contraintes :
choix d´une des machines connectées uniquement

Messages de sortie :
numéro de JBL liée à cette machine

SAUVETAGE RESULTATS

Objectifs : sauvetage des résultats normaux et acceptés dans la BD

Messages d´entrée :
l´ensemble des résultats

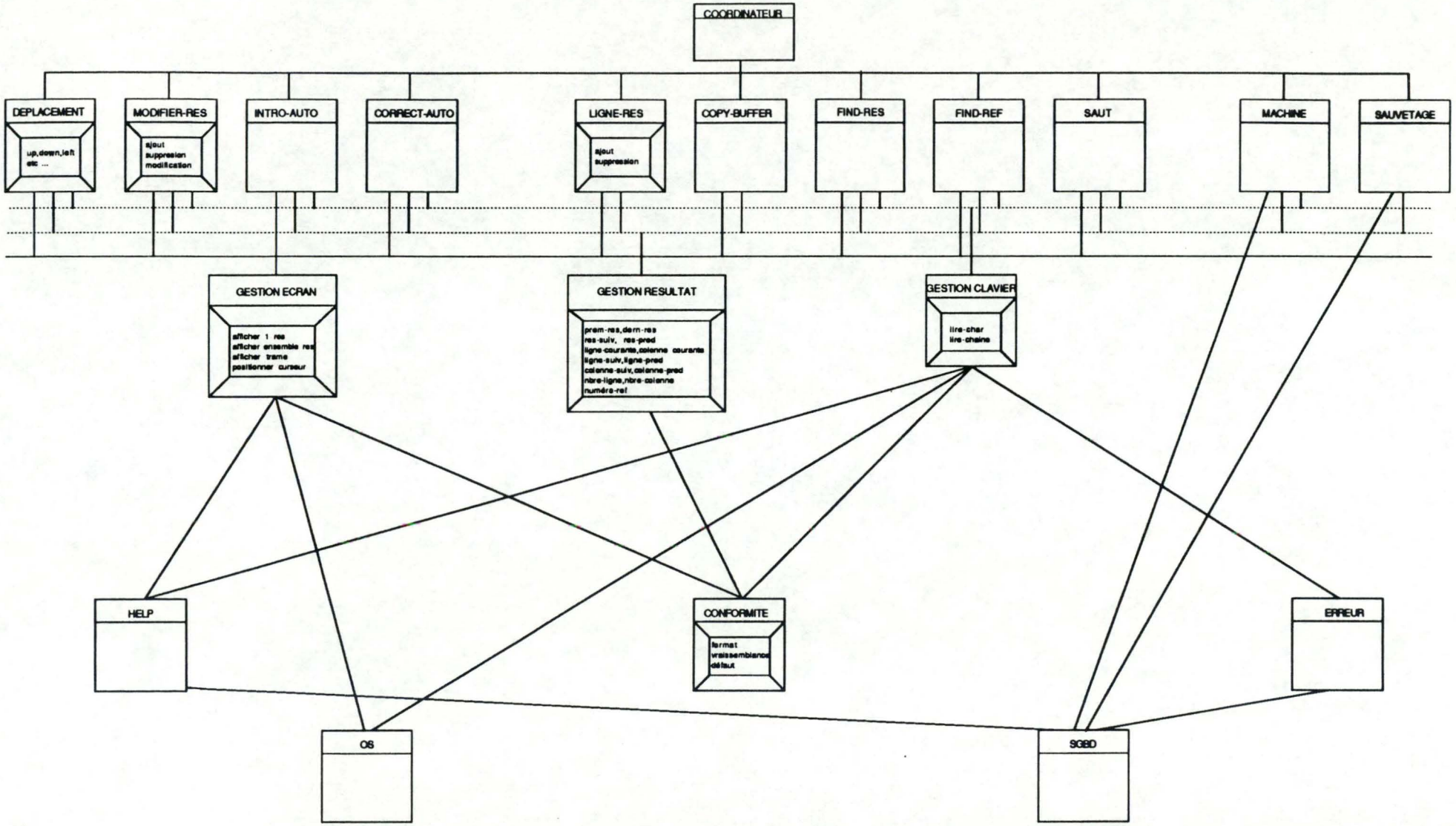
Contraintes : /

Messages de sortie :
"Attente S.V.P."
"Problème lors du sauvetage, vérifiez et recommencez"

Développement d'une architecture logicielle

Structuration et hiérarchisation

La relation "utilise" est employée dans cette hiérarchisation.


COORDINATEUR
DEPLACEMENT
up,down,left
etc ...
MODIFIER-RES
ajout
suppresion
modification
INTRO-AUTO
CORRECT-AUTO
LIGNE-RES
ajout
suppression
COPY-BUFFER
FIND-RES
FIND-REF
SAUT
MACHINE
SAUVETAGE
GESTION ECRAN
afficher 1 res
afficher ensemble res
afficher trame
positionner curseur
GESTION RESULTAT
prem-res,dern-res
res-suiv, res-pred
ligne-courante,colonne courante
ligne-suiv,ligne-pred
colonne-suiv,colonne-pred
nbre-ligne,nbre-colonne
numéro-ref
GESTION CLAVIER
lire-char
lire-chaine
HELP
CONFORMITE
format
vraisemblance
défaut
ERREUR
OS
SGBD

Spécification externe de chaque module

Module de niveau 6 :
Spécifications décrites dans la spécification et description des fonctions

Module de niveau 5 :

GESTION ECRAN :
module de type donnée, visible uniquement par ses primitives

AFFICHER-RES

Données : page courante
type de donnée
résultat courant
position courante

Fonction :
affiche le résultat à l'écran à sa place correcte sur l'écran et suivant le résultat et les valeurs de vraisemblances liées à cette analyse
Rappel : gras : résultat en dehors des normes
clignotant : résultat superflu
'-----' : résultat manquant
normal : résultat normal
' ' : sans résultat et non-manquant

AFFICHER ENSEMBLE DE RESULTATS

Données : page courante

Résultats : /

Fonction :
généralisation de la fonction précédente à une page de résultats

POSITIONNER CURSEUR

Données : position courante
nouvelle position courante

Résultats : /

Fonction :
supprime le curseur de sa position courante et le place à la nouvelle position. Si une des positions n'existe pas, le résultat n'est pas assuré

AFFICHER TRAME

Données : page courante

Résultats : /

Fonction :
affiche les noms des analyses et les numéros de référence de la page courante

GESTION CLAVIER :

module de type donnée, visible uniquement par ses primitives.

LIRE CHAR

Données : /

Résultats : char : caractère affichable
spec-char : caractère composé ou spécial

Fonction :
lit au clavier soit un caractère affichable (résultat = char et spec-char = NUL) soit un caractère non-affichable (résultat = spec-char et char = chr(0)). De cette manière, TOUTES les touches du clavier sont définies.

LIRE CHAINE DE CHAR

Données : position écran
nombre maximum de caractères

Résultats : chaîne de caractères

Fonction :
lit une chaîne de caractères à partir du clavier et de longueur inférieure ou égale au nombre maximum de caractères

GESTION RESULTATS :

module de type donnée, uniquement visible par ses primitives

NUM-REF

Données : numéro de ligne (i)

Résultats : numéro de référence

Fonction :

donne le numéro de référence de la ligne 'i'. Si cette ligne n'existe pas, le résultat n'est pas assuré.

NBRE-LIGNES (NBRE-COLONNES)

Données : /

Résultats : nombre de lignes (colonnes)

Fonction :

donne le nombre de lignes (colonnes) associées au "tableau" de résultats

COL-SUIV (LIG-SUIV)

Données : numéro de colonne (ligne) courante

Résultats : colonne (ligne)

Fonction :

donne la colonne (ligne) suivant la colonne (ligne) courante. S'il n'existe pas de colonne (ligne) suivante, le résultat n'est pas assuré

COL-PRED (LIG-PRED)

Données : numéro de colonne (ligne) courante

Résultats : colonne (ligne)

Fonction :
donne la colonne (ligne) précédant la colonne (ligne) courante. S'il n'existe pas de colonne (ligne) précédente, le résultat n'est pas assuré

PREM-COL (PREM-LIG)

Données : /

Résultats : colonne (ligne)

Fonction :
donne la première colonne (ligne) du tableau de résultats

DER-COL (DER-LIG)

Données : /

Résultats : colonne (ligne)

Fonction :
donne la dernière colonne (ligne) du tableau de résultats

AJOUT-RES (SUPP-RES, MOD-RES)

Données : position courante
nouvelle valeur de résultat

Résultats : résultat

Fonction :
ajoute (supprime, modifie) la nouvelle valeur de résultat à la position courante (remarque : si le résultat est supprimé, sa valeur = -1)

COL-COURANTE (LIG-COURANTE)

Données : /

Résultats : numéro de colonne (ligne)

Fonction :
retourne le numéro de colonne (ligne) courante

RES-SUIV (RES-PRED)

Données : numéro de ligne
numéro de colonne

Résultats : résultat

Fonction :
donne le résultat suivant (précédent) le résultat courant. Si le résultat n'existe pas, le résultat de la fonction n'est pas assuré

PREM-RES (DERN-RES)

Données : numéro de ligne (i)

Résultats : résultat

Fonction :
donne le premier (dernier) résultat de la ligne 'i'

PREM-RES-BUFF (DERN-RES-BUFF)

Données : buffer de ligne

Résultats : résultat

Fonction :
donne le premier (dernier) résultat du buffer de ligne

RES-BUFF-SUIV (RES-BUFF-PRED)

Données : buffer de ligne
résultat courant du buffer

Résultats : résultat

Fonction :
donne le résultat suivant (précédent) le résultat courant du buffer de ligne. S'il n'existe pas de résultat suivant (précédent), le résultat de la fonction n'est pas assuré.

Modules de niveau 4 :

Module HELP-ERREUR

Objectifs : afficher des messages d'aide ou d'erreur

Messages d'entrée : numéro

Contraintes : le numéro de message doit exister

Message de sortie : affiche un message

CONFORMITE DE DONNEE :

module de type donnée, visible uniquement par ses primitives

Remarque : la description des données est intégrée dans ce module

FORMAT

Données : type de résultat (d'analyse)
résultat

Résultats : booléen

Fonction :

définit si le résultat respecte bien le type de résultat

VALEUR-VRAISEMBLANCE

Données : type de résultat
résultat

Résultats : booléen

Fonction :
définit si le résultat est compris dans les normes de vraisemblance associé à ce type d´analyse

VAL-DEFAUT

Données : type de résultat

Résultats : résultat

Fonction :
donne la valeur par défaut de ce type de résultat

Pourquoi une telle découpe ?

A maintes reprises, la découpe résultante fut très compliquée et ne respectait pas la découpe "utilise" en tous points. Cette découpe est à la fois fort simple et correcte, ce qui facilite la compréhension et l'implémentation des modules.
Pourquoi utiliser des modules de données pour

GESTION ECRAN : Grâce à ce module de données, si un jour on pense travailler sur des terminaux de type différent, il suffira de modifier ce module (ou de le compléter s'il existe la posibilité de faire une reconnaissance dynamique du type de terminal)

GESTION CLAVIER : même remarque

GESTION RESULTAT : La structure de données qui gère la mémorisation des résultats peut être de type très différent. La structure sera choisie en fonction de paramètres tels que rapidité, coût, fiabilité, ... Si un de ces points devait changer radicalement, on devrait modifier la structure et les modifications seraient cantonnées à ce module.

Spécification interne des modules

DEPLACEMENT : affichage du plus grand nombre de résultats possible.
MODIFIER-RES : /
RECHERCHE-VALEUR : /
INTRO-AUTOMATIQUE : possibilité de stopper la fonction à tout instant (le message de sortie est alors modifié).
CORRECTION-AUTOMATIQUE : possibilité de stoppper la fonction à tout moment.
AJOUT-DEL-LIGNE DE RES : /
COPY-BUFFER : /
RECHERCHE-NO-REFERENCE : /
SAUT : /
CHOIX MACHINE-ANALYSE : choix par sélection uniquement. Si la JBL associée est vide, alors un message d'avertissement est proposé et le programme stoppé.
SAUVETAGE RESULTATS : /

AFFICHER-RES : le résultat se trouve au centre de la colonne d'analyse.
AFFICHER ENSEMBLE DE RESULTATS : recherche de rapidité.
POSITIONER CURSEUR : /
AFFICHER TRAME : /
LIRE CHAR : les caractères spéciaux seront définis comme type en début de programme.
LIRE CHAINE DE CHAR : Cfr les spécifications définies par monsieur Roulin.
NUM-REF : s'il n'existe pas alors on transmet le numéro de référence de l'élément courant.
NBRE-LIGNES (NBRE-COLONNES) : résultat toujours suppérieur à 0.
COL-SUIV (LIG-SUIV) : le résultat existera toujours (mais pas necessairement correct).
COL-PRED (LIG-PRED) : le résultat existera toujours (mais pas nécessairement correct).
RES-SUIV (RES-PRED) : le résultat existera toujours (mais pas

necessairement correct).
PREM-COL (PREM-LIG) : /
DER-COL (DER-LIG) : /
AJOUT-RES (SUPP-RES, MOD-RES) : /
COL-COURANTE (LIG-COURANTE) : /
PREM-RES (DERN-RES) : /
PREM-RES-BUFF (DERN-RES-BUFF) : /
RES-BUFF-SUIV (RES-BUFF-PRED) : le résultat existera toujours.

HELP-ERREUR : si les fichiers sont inaccessibles alors on affiche "no comments" pour le help et "undefined error" pour les erreurs. Dans le cas du help, il peut être composé de plusieur lignes accessibles de manière cyclique. Dans l'un et l'autre cas, la traduction en langue étrangère peut se faire sans trop de frais (sans compilation,...)

FORMAT : /
VALEUR-VRAISEMBLANCE : /
VAL-DEFAUT : /

Validation du logiciel

Conception d'un plan de tests black box des modules

MODIFIER-RES :

- ajouter, supprimer, modifier un résultat; ce qui était attendu.
- ne pas respecter le format d'entrée; refus
- résultat en dehors des normes de vraisemblance; apparait en gras
- supprimer un résultat nécessaire; apparait '----'
- supprimer un résultat superflu; rien ne doit apparaître
- modifier, ajouter un résultat superflu; apparait en clignotant
- supprimer un résultat manquant; aucune modification
- help

RECHERCHE-VALEUR :

- valeur avec format correct et existant après la position courante; afficher le résultat recherché
- format incorrect; message d'erreur
- valeur n'existant pas après la position courante; message d'erreur
- help

INTRO-AUTOMATIQUE :

- résultats manquants après la position courante; introduction de tous ces résultat manquants
- sans résultat manquant; message d'avertissement
- introduction de résultat sans respect du format; message d'erreur et refus
- introduction d'un résultat en dehors de ses normes de vraisemblance; affichage en gras
- ne rien introduire; aucune modification
- arret de la fonction; message d'avertissement et fonction stoppée
- help

VERIFICATION-AUTOMATIQUE

- sans valeur anormale; message d'avertissement et fin immédiate de la fonction
- un tableau composé de résultats anormaux uniquement; passage en revue de tous les résultats
- acceptation d'un résultat anormal;
- passage au résultat anormal suivant sans modification; aucune modification

AJOUT-DEL-LIGNE DE RES :

- ajout en première (dernière) ligne;
- suppression de la première (dernière) ligne;

COPY-BUFFER :

vérifier que c'est bien le buffer qui est recopié

CHERCHE-NO-REFERENCE :

- numéro de référence existant; cfr contraintes
- numéro de référence n'existant pas; message d'erreur
- mauvais format du numéro de référence; message d'erreur

SAUT :

- première, dernière ligne, première, dernière colonne;
- numéro de ligne qui n'existe pas; dernière ligne
- numéro de colonne qui n'existe pas; dernière colonne
- mauvais format de numéro de ligne ou de numéro de colonne; message d'erreur

CHOIX MACHINE-ANALYSE : /

SAUVETAGE RESULTATS :

- vérifier les problèmes de concurrence; attente et message d'erreur
- les problèmes de fichier; message d'erreur

AFFICHER-RES :

résultat superflu; clignotant

résultat pathologique; gras

résultat manquant; '-----'

résultat normal;

AFFICHER-ENSEMBLE DE RES :

- page complète;
- page vide;
- page normale;

POSITIONNER CURSEUR :

- 2 positions correctes;
- position non-exacte; plantage pur et simple
- 2 positions identiques;

AFFICHER TRAME : /

LIRE CHAR :

- essayer pour tous les caractères affichables et non-affichables

LIRE CHAINE DE CHAR :

- chaine vide;
- nombre de caractères inférieur à 0; arrêt immédiat
- normal;

NUM-REF : /

NBRE-LIGNES (NBRE-COLONNES) : /

COL-SUIV (LIG-SUIV) :

- colonne (ligne) courante = dernière colonne (ligne); résultat incorrect mais sans plantage.

COL-PRED (LIG-PRED) :

- colonne (ligne) courante = première colonne (ligne); résultat incorrect mais sans plantage.

RES-SUIV (RES-PRED) :

- résultat courant est le dernier (premier); résultat incorrect mais sans plantage

PREM-COL (PREM-LIG) : /

HELP-ERREUR :

- numéro n'existant pas; pas d'affichage d'erreur

FORMAT :

- format correct; true
- format incorrect; false;

VALEUR-VRAISEMBLANCE :

- résultat dans les valeurs; true
- résultat en dehors des valeurs; false

VAL-DEFAUT : /

Conception de l'architecture physique

Implémentation

L'implémentation va bouleverser de manière importante la proposition faite ci-avant. En effet pour des raisons de performance (rapidité), principal problème à résoudre, les modules de données vont disparaître (les structures de données vont devenir globales), des relations import/export vont être nécessaire entre modules (gestion écran - gestion résultats).

Intégration

Les tests d'intégration seront du type incrémental. Le système d'exploitation, la base de données et le module de conformité ont déjà été testés. Le module Help-erreur est très facile à tester. Il ne reste plus que les modules de niveau 5 et 6 dont l'intégration peut poser problème.

Sous-système

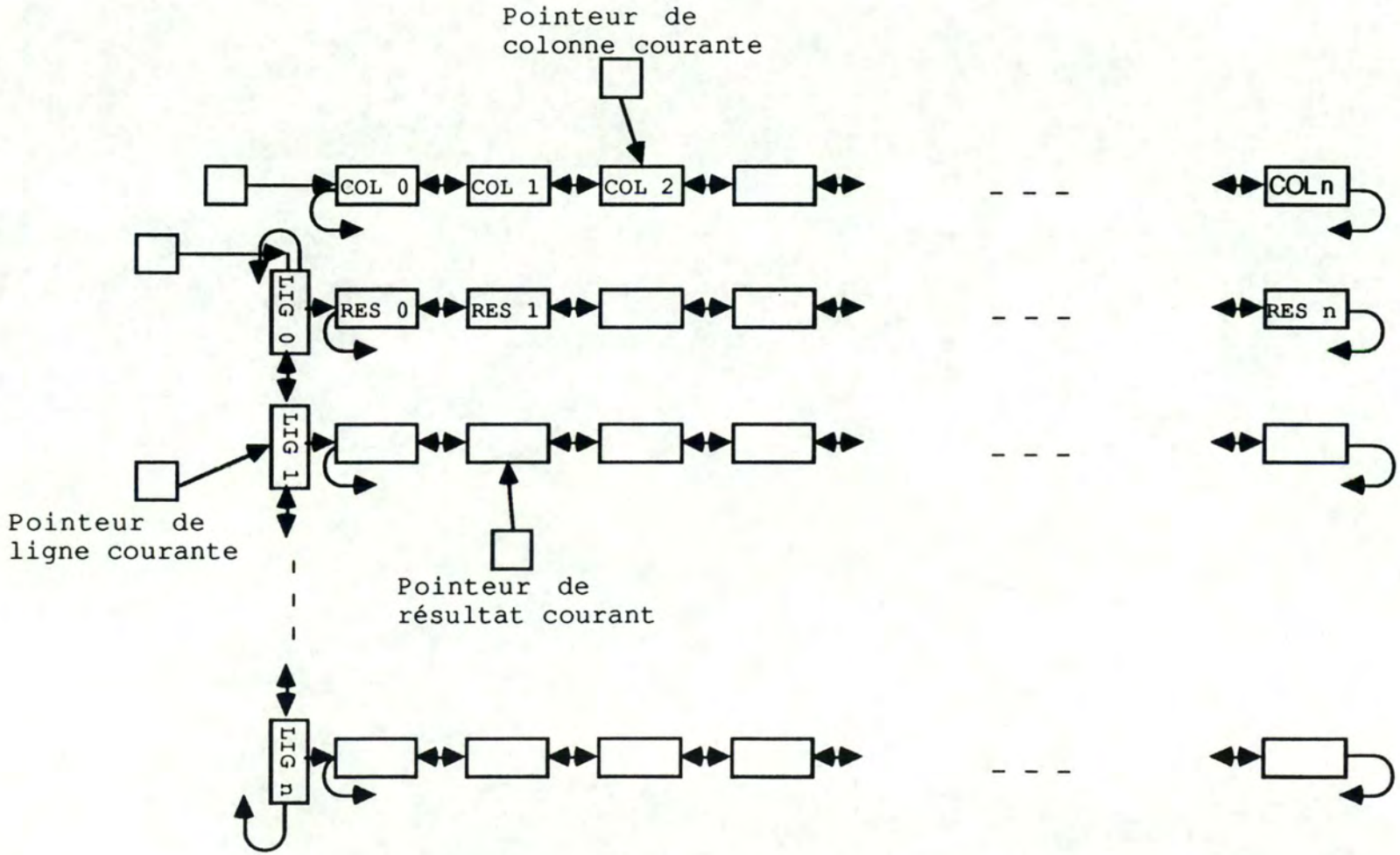
Les sous-systèmes utiles sont facile à déterminer: le premier sera constitué du module de déplacement, du module d'initialisation et de choix de machine d'analyse, et des modules de niveau 5. Ensuite, en ajoutant à ce premier sous-système une des fonctions du niveau 6, nous pouvons trouver un nouveau système utile.

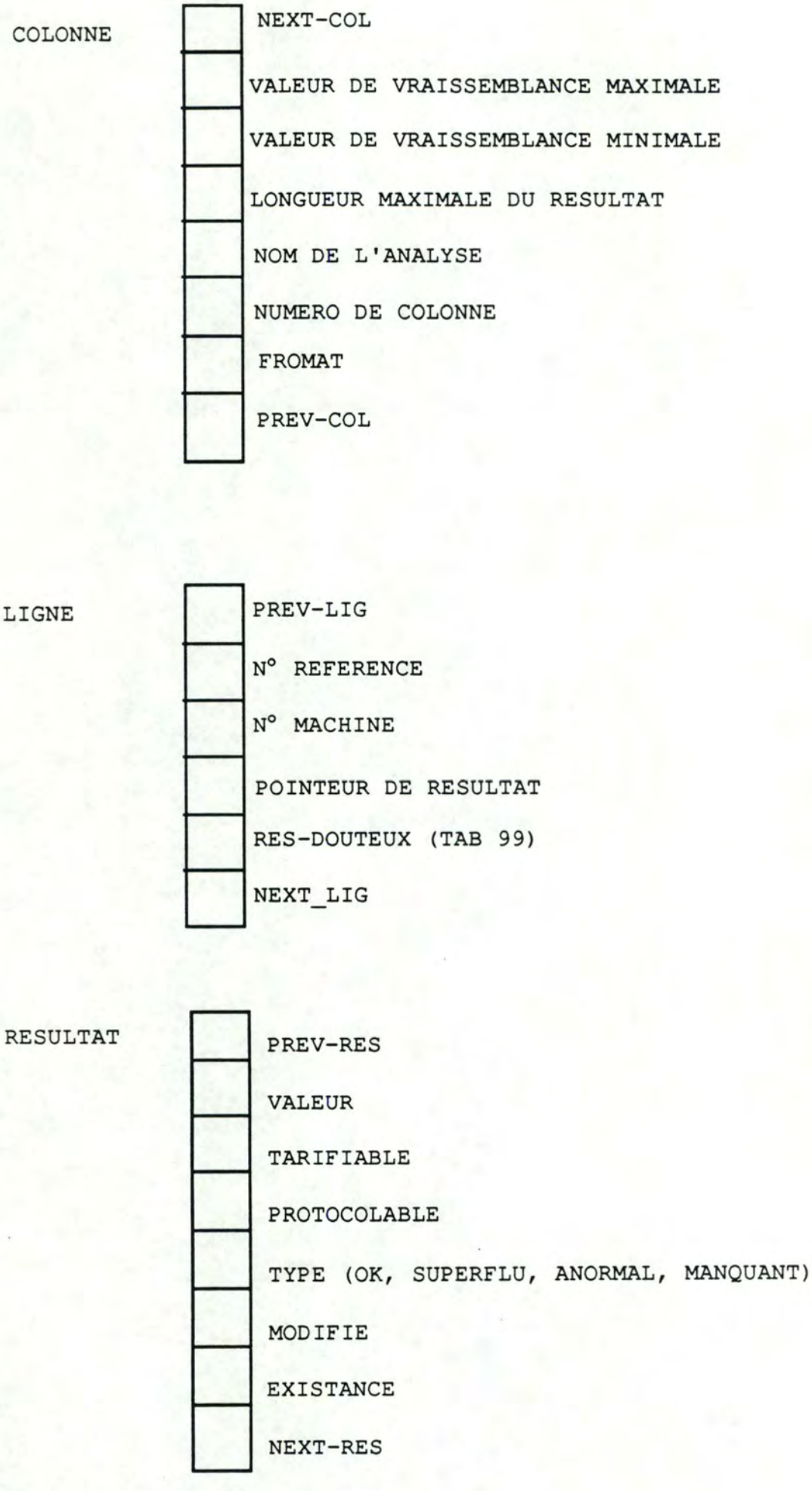
Performances

Des tests de performances seront effectués tout au long de l'implémentation physique. Il est possible que des changements soient effectués en cours de réalisation.

Schéma de la structure de données

Représentation des données du
programme d'acceptation




COLONNE
NEXT-COL
VALEUR DE VRAISSEMBLANCE MAXIMALE
VALEUR DE VRAISSEMBLANCE MINIMALE
LONGUEUR MAXIMALE DU RESULTAT
NOM DE L'ANALYSE
NUMERO DE COLONNE
FROMAT
PREV-COL
LIGNE
PREV-LIG
N° REFERENCE
N° MACHINE
POINTEUR DE RESULTAT
RES-DOUTEUX (TAB 99)
NEXT_LIG
RESULTAT
PREV-RES
VALEUR
TARIFIABLE
PROTOCOLABLE
TYPE (OK, SUPERFLU, ANORMAL, MANQUANT)
MODIFIE
EXISTANCE
NEXT-RES

PROGRAMME DE CONNECTION

Spécification du programme de connection

Les trois fonctions du programme sont

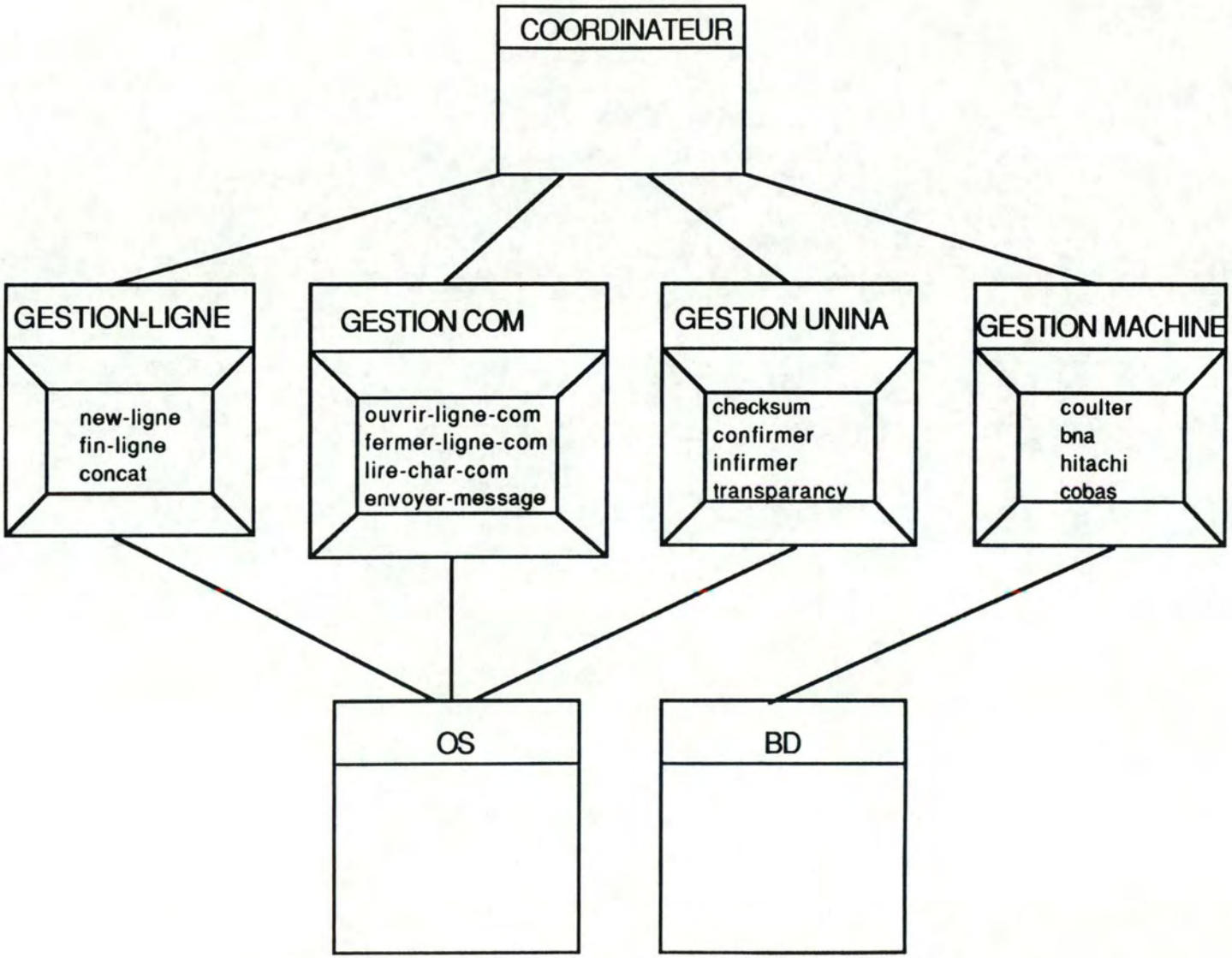
- Mise en oeuvre des protocoles de connection avec le systeme UNINA. Ceci se compose de la vérification de transfert, et d'un système d'acceptation ou de refus des données.
- Conversion des formats des machines d'analyses en un format unique.
- Sauvetage des résultats et autres données dans un fichier en fonction d'éléments tels que numéro de séquence, type d'analyse et type de machine.

Il n'existe aucune interaction avec l'utilisateur; le seul évenement extérieur pris en compte est la survenance d'un message en provenance du concentrateur UNINA.

Développement d'une architecture logicielle

Structuration et hiérarchisation

La relation "utilise" sera employée pour la hiérarchisation.


COORDINATEUR
GESTION-LIGNE
new-ligne
fin-ligne
concat
GESTION COM
ouvrir-ligne-com
fermer-ligne-com
lire-char-com
envoyer-message
GESTION UNINA
checksum
confirmer
infirmer
transparancy
GESTION MACHINE
coulter
bna
hitachi
cobas
OS
BD

Spécifications externes de chaque module

Modules de niveau 6 :

GESTION LIGNE :

module de type données, visible uniquement par ses primitives.

Nom : NEW LIGNE

Données : /

Résultats : ligne

Fonction : donne une ligne vide

Nom : FIN LIGNE

Données : ligne
caractères de fin de ligne

Résultats : booléen

Fonction : définis si oui ou non les derniers caractères de la ligne sont des caractères de fin de ligne

Nom : CONCAT

Données : ligne
caractère

Résultats : ligne

Fonction : ajoute le caractère en fin de ligne.

GESTION UNINA :

module de type données, visible uniquement par ses primitives.

Nom : CHECKSUM

Données : ligne

Résultats : booléen

Fonction : applique la fonction de checksum proposé par UNINA. Si ce checksum se révele être identique à celui transmis, la fonction retourne true, false autrement

Nom : TRANSPARENCY

Données : ligne

Résultats : ligne

Fonction : procédure proposée par la firme UNINA pour transmettre tous les caractères. Le système UNINA devient transparent pour tous les caractères.

Nom : CONFIRM

Données : /

Résultats : /

Fonction : transmet au système UNINA le message de confirmation de transfert correct.

Nom : INFIRM

Données : /

Résultats : /

Fonction : transmet au système UNINA le message de transfert incorrect

GESTION COMMUNICATION :

module de données, visible uniquement par ses primitives.

Nom : OUVRIR-LIGNE-COM

Données : /

Résultats : /

Fonction : initialise la porte pour la communication.

Nom : FERMER-LIGNE-COM

Données : /

Résultats : /

Fonction : libère la porte de communication.

Nom : LIRE-CHAR-COM

Données : /

Résultats : caractère

Fonction : lit un caractère sur la porte précédemment ouverte.

Nom : ENVOY-MESSAGE-COM

Données : message

Résultats : /

Fonction : transmet un message par la porte de communication précédement ouverte.

GESTION MACHINE :

module de données, visible uniquement par ses primitives.

Nom : COULTER (BNA,HITACHI,COBAS)

Données : ligne

Résultats : résultat

Fonction : en fonction de la ligne, des formats et des contraintes de chaque machine, on transmet le résultat dans un fichier.

Pourquoi une telle découpe ?

Tous ces modules de données permettent de cacher le plus de représentations de données possibles.
Dans le cas

- de changement de représentation de ligne
- de modification de type de communication
- de remplacement du système UNINA par un autre (ou système bidirectionnel)
- d'ajout d'une nouvelle machine d'analyses

un seul module doit être modifié.

Validation du logiciel

Conception d'un plan de tests black box des modules

GESTION DE LIGNE :

NEW LIGNE :

- utiliser cette fonction et vérifier que la ligne est effectivement vide

FIN LIGNE :

- ligne vide; booléen à faux
- ligne composée uniquement de carctères de fin de ligne; booléen à vrai
- ligne de longueur maximale et sans caractère de fin de ligne; booléen à faux

CONCAT :

- ligne vide; ligne composé d'un seul caractère
- ligne de longeur maximale; le dernier caractère est remplacé
- caractère particulier (fin de ligne, CR, ...); ligne completée normalement

GESTION UNINA :

CHECKSUM :

- ligne et checksum transmis corrects; booléen à vrai
- ligne et checksum transmis non corrects; booléen à faux

GESTION COMMUNICATION :

OUVRIR-LIGNE-COM :

- ligne de communication utilisée par quelqu´un d´autre; résultat non assuré
- ligne de communication non allouée; initialisation correcte

LIRE-CHAR-COM :

- lecture d´un caractère sur une porte non ouverte; résultat non assuré
- aucun caractère transmis; lecture en attente

ENVOY-MESSAGE-COM :

- message; vérifier que le message est correctement transmis

GESTION MACHINE :

COULTER (BNA,HITACHI,COBAS)

- vérifier chaque contrainte
- ligne incorrecte au point de vue format; résultat non transmis
- ligne vide; aucun résultat transmis

Conception de l´architecture physique

Implémentation

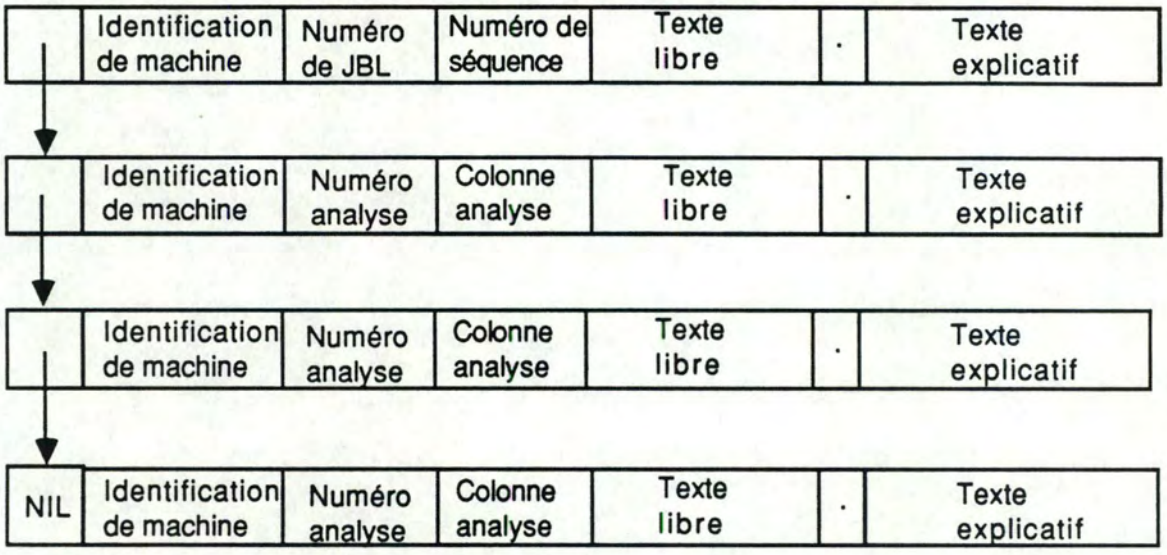
L´implémentation physique respecte en tous points la découpe en module.

Intégration

Les tests d´intégration seront incrémentals et il n´existe qu´un seul sous-système utile: la totalité.

Schéma de la structure de données

Représentation des données du
programme de connection



CHAPITRE IV
REALISATION INFORMATIQUE

REALISATION INFORMATIQUE

Conscients du peu d'intérêt porté généralement à une analyse fouillée de l'implémentation proprement dite, nous en faisons grâce au lecteur; les plus avertis trouveront cependant les textes des programmes en annexe.

Nous jugeons toutefois capital d'étayer dans un premier temps nos principaux choix informatiques -choix à priori- et de soulever dans un second temps quelques objections concernant ces choix -réflexions à postériori-.

Choix à priori.

Rapidité

L'essence même de l'application, nous l'avons dit, n'est autre que la rapidité (premier axe du programme). Ce souci de rapidité nous a conduits d'emblée à un certain nombre de choix et ce à divers niveaux.

Au niveau de la gestion d'écran:

refus d'utiliser SMG -un système de gestion d'écran disponible-. Nous lui substituerons notre propre système qui doit sa rapidité à des outils de 'bas niveau'. Ce fut notre première victoire sur le temps. Dès lors, nous étendrons cette manière de procéder à la plus grande partie de notre programme en exploitant au maximum des fonctions de bas niveau disponibles dans diverses librairies du système.

Au niveau des appels de procédure:

limitation des appels de procédure abusifs tout en tâchant de sauvegarder un maximum de clarté. On constate en effet qu'une utilisation abusive des appels procéduraux entravent significativement les performances de vitesse alors qu'à première vue, le temps d'un seul appel apparaît comme négligeable.

Au niveau des accès aux données.

Nous favoriserons donc au maximum la rapidité -les temps d'accès mémoire n'ayant pas leur pareil- au détriment d'une gestion plus rationnelle de la mémoire. Toutefois, ce n'est pas la mémoire qui nous manque ...

Flexibilité

La survie de toute application dépendant dans la plupart des cas de ces facultés d'adaptation aux situation nouvelles -prévisibles ou non-, nous veillerons soigneusement à doter la nôtre d'un capital suffisant de flexibilité (second axe de notre programme)

Découpe en modules

Nous nous efforcerons de nous conformer à la découpe en modules dégagée par le chapitre précédent. Il est vrai qu'il faudra quelquefois nous résoudre à trouver des compromis vitesse-souplesse mais c'est le prix à payer pour éviter que notre application ne tombe trop rapidement en désuétude!

Fichiers intermédiaires

Nous sauvegarderons erreurs et commentaires dans des fichiers de type texte , ce qui nous permettra de les modifier en employant un simple éditeur de textes.
Les résultats transmis par les machines d´analyses seront sauvés quant à eux, dans un fichier séquentiel indexé de facon à, d´une part, pouvoir accéder à tout article du fichier à l´aide d´une clé et d´autre part à respecter une certaine homogéinité avec le programme existant dont les fichiers sont précisément de ce type.

Les réflexions à postériori

Aucun programme, aussi bien pensé puisse-t-il être, n'est parfait. Nous ne prétendrons donc pas échapper à cette assertion. Passer sous silence les améliorations qui s'imposent ou qui s'avèrent souhaitables, adoucir au maximum les critiques de notre propre travail nous a semblé un faux fuyant peu rentable. Dans un but constructif nous procéderons de la technique contraire.

Les améliorations

Précisons d'emblée que l'énumération ci-jointe des améliorations n'est probablement pas exhaustive; ce sont celles qui de diverse manières nous sautèrent aux yeux.

Adjonction de commentaires spécifiques aux résultats

Le but de cette amélioration est de commenter tout résultat quant à son exécution ou sa validité. Il est clair que ces notices ne sont pas générées par la machine elle-même et c'est pourquoi le commanditaire de l'application refusa cette possibilité. Nous regrettons que les utilisateurs ressentent ce manquement dans quelques cas.

Elargissement des normes de vraisemblance

Rappelons que pour valider un résultat nous testons si oui ou non sa valeur est une valeur vraisemblable. La marge de manoeuvre trop étroite entre normes de vraisemblance et normes de référence conduit fréquemment à nombre de tests inutiles. Une redéfinition plus large de ces normes accroîtrait certainement l'efficacité de l'application sans en entraver la fiabilité. Cette modification malheureusement ne nous revient pas.

Utilisation d'un programme de transfert en batch

Cette suggestion vise à réduire fortement, sinon à néant le temps d'attente pendant le transfert dans la B.D. des résultats du programme d'acceptation, temps durant lequel l'utilisateur ne peut que prendre son mal en patience.
La solution actuelle -transfert des résultats dans des fichiers séquentiels indexés à clés multiples- pourrait être avantageusement remplacée par l'emploi d'un fichier temporaire de type séquentiel -donc très rapide- et d'un programme batch prenant en charge le transfert tandis que la main serait rendue au courageux utilisateur.
Cette nouvelle solution serait, d'après nous, près de 30 fois plus rapide; une grande prudence s'impose toutefois: les problèmes de concurrence sont aigus.

Création d'un fichier journal et mise à jour.

Lorsque survient un problème résultant de l'omission de la programmation d'une analyse, les démarches à suivre pour rectifier le tir coûtent trop de temps: programmation de l'analyse,d'abord, introduction du résultat ensuite. Dans le cas particulier de l'acceptation d'un résultat "superflu", la valeur du résultat, le nom de l'analyse et le numéro de référence de l'échantillon sont sauvés dans un fichier journal. Ainsi, un programme lancé en parallèle peut reprendre ces cas particulier, créer la demande correspondant et sauver les résultats sans aucune manipulations manuelle. Si le fichier journal est dès à présent crée, le programme de mise à jour ne demande qu'à exister.

Les critiques

D'un point de vue purement technique, nous pensons que quatre objections pourraient nous être avancées.

Texte confus de certaines fonction

Nous accepterons volontiers cette remarque. en effet, l'utilisation des outils de bas niveau visant à accroître encore et toujours la rapidité peut rebuter à la première lecture. Cependant, nous nous sommes efforcés -dans l'ensemble- et chaque fois que cela c'est avéré possible de conserver un minimum de clarté. Trouver un meilleur compromis clarté/rapidité est peu vraisemblable.

Utilisation de la mémoire peu parcimonieuse.

Ce reproche s'adresse à notre gestion dynamique de la mémoire par le biais de pointeurs. Nous reconnaissons que, quelque soit le nombre d'analyses demandées nous créons imperturbablement le même nombre de résultats. Nous restons cependant convaicu que les risques de fréquents gaspillages mémoire sont minimes si l'on s'en réfère aux statistiques.

Choix d'un fichier texte peu pertinent pour les données communes au programme d'acceptation et de connection

Nous avons nous-même remis en question ce choix à maintes reprises. La gestion de ce fichier est trop lourde et le gain de flexibilité ne nous apparait plus -à postériori- faire le poids.

Programme d'acceptation non conforme aux besoins de l'utilisateur.

Cette critique nous vient implicitement des utilisateurs actuels de ce programme: ils le sous-utilisent à notre grand désarroi: nos tentatives de formations, la rédaction d'un mode d'emploi détaillé, un synoptique des principales fonctions n'y changèrent rien.

Critiquer n'étant pas réfuter, nous terminerons ce chapitre sur les quelques notes positives suivantes:

Flexibilité du système

Une simple modification des phases d'initialisation et de terminaison suffit à transformer notre programme en un programme d'introduction de résultats soit par JBL soit par type de résultat ou encore en un logiciel d'acceptation des demandes d'analyses dans le cadre d'un ONLINE bidirectionnel.

Rencontre globale des besoins de l'utilisateur

Un sondage parmi les utilisateurs sur d'éventuelles modifications à apporter au programme nous révele qu'aucun apport n'était souhaité. Nous avions en face de nous, des utilisateurs heureux!

CHAPITRE V

MISE EN PLACE DU PROGRAMME

MISE EN PLACE DU PROGRAMME

Les quelques réflexions qui suivent font suite à un cours et une discussion avec le Père Berleur au sujet des difficultés rencontrées lors de l'implantation d'un système informatique dans une entreprise. Nous ne tentons en aucune manière de les analyser ou de les interpréter. Elles sont le fruit de notre maigre expérience et sont à apprécier par les principaux membres du laboratoire et de ses informaticiens. C'est le seul but de ce chapitre.

Allo!...Kissinger ?

Diplomatie... L'introduction de la mise ONLINE ne s'est pas toujours bien déroulée. Nombre de réflexions du genre "doucement hein, il nous faut encore du boulot" furent proférées. Il est alors important de convaincre les principaux acteurs que l'intêret de l'informatisation se porte sur la qualité du travail plutôt que sur la rentabilité. Une qualité indispensable pour un informaticien: la DIPLOMATIE.

Ergonomie <> présentation

Les utilisateurs habituels d'un programme ne portent que peu d'intérêt à la beauté de la présentation d'un écran. Ils demandent simplicité et facilité d'emploi avant tout. Informaticiens, ne sacrifiez pas rapidité et facilté au profit de la présentation !

Conseil ?

Depuis que nous avons jeté notre dévolu sur l'informatique, un des conseils les plus souvent prodigués est: "il faut se mettre à la place de l'utilisateur". Ce conseil n'est pas le meilleur qui nous fut donné. Il est impossible de raisonner, et travailler comme quelqu'un d'autre. Pour preuves, nous vous donnons deux exemples.

A chacun son boulot

Les résultats des tests de rapidité que nous avons effectués nous semblaient suffisants. Et pourtant... La vitesse de frappe d'une secrétaire déchaînée est incomparable à la notre.

A chacun son raisonnement

Les efforts exigés par l'adjonction de certaines fonctions furent parfois inutiles. En effet, certaines de ces fonctions n'ont jamais été employées. Par contre, certains points qui me semblaient futiles posèrent parfois de gros problèmes aux utilisateurs.

Plutôt que d'essayer de se mettre à la place de l'utilisateur, une collaboration active avec celui-ci peut être beaucoup plus rentable. Les possibilités de prototypages dans les languages de quatrième génération permettent cette collaboration tout au long du développement du logiciel.

CONCLUSION

CONCLUSIONS

Objectifs ? En vue !
Distance ? Quelques lignes supplémentaires de programme !

La réalisation devrait satisfaire complètement la sphère directrice. En effet, la mise ONLINE s´est effectuée avec succès à la date prévue. Depuis, peu de problèmes sont apparus. Voilà quelques mois que le programme est utilisé sans erreurs détectées. Les quelques modifications proposées dans un chapitre précédent devraient être apportées et l´utilisation étendue à un plus grand nombre de machines (si c´est possible). Ainsi, les connections révèleraient leurs étonnantes possibilités et à la limite, la nécessité de leur utilisation.

Bilan : équilibré !

La réalisation de ce projet institue deux bénéficiaires: le laboratoire et moi-même. Les bénéfices étaient équitables.

L'informatisation apporte à l'entreprise :

- nouveauté : de nouvelles possibilités de traitements sont apparues
- rapidité : maître mot dans cette réalisation
- fiabilité : le risque d'erreur d'encodage est beaucoup plus faible qu'une introduction habituelle de résultats
- facilité : une vérification de résultats est beaucoup plus simple qu'une introduction
- rentabilité : en augmentant rapidité et fiabilité, la rentabilité est accrue.

Le projet d'informatisation nous a apporté deux choses principales :

- augmentation (constitution) de notre capital expérience
- réalisation d'un mémoire.

Et maintenant ...

Ce mémoire devrait apporter une aide précieuse dans la réalisation de nouvelles connections. Ces quelques pages serviront également de documentation pour la maintenance des produits réalisés.

Dans le futur...

Le choix technologique fait par le laboratoire ne se révèle pas aussi prometteur qu'espéré. Ce mémoire pourrait inciter la direction à s'interroger sur le bien fondé de nouvelles mise ONLINE basées sur le système UNINA. D'autres solutions technologiques n'existent-elles pas ?

BIBLIOGRAPHIE

BIBLIOGRAPHIE

==========

Conception assitée des applications informatiques (Etude d'opportunité et analyse conceptuelle)
F.Bodart et Y.Pigneur
Masson et presse universitaire de Namur.

Fundamentals of clinical chemistry
Wendel T. Caraway
Edition Sauders

Database design in Entity-Relationship approach
P. MOSANTO
Edition POSEIDON

Data transmission analys, design, application
Dogan Tugal and Osman Tugal
Edition Mac-Graham-hill

Technical aspect of data communication
John E Mc Namara
Edition digital

Idata.doc
L. Claes
(Photocopie firme UNINA)

UNINA BIP
L. Claes
(photocopie firme UNINA)

Système LDS
?
(Photocopie firme PGP)

Dictionnaire de l´informatique
P.Morvan
Larousse

Documentation des machines d´analyses

- Hitachi 705
- Hitachi 704
- Hitachi 737
- Cobas Bio
- Cobas Fara
- Cobas Mira
- BNA
- Coulter S7
- Sebia Profil
- Monarch
- Ependorf Iris

TABLE DES MATIERES

TABLE DES MATIERES

Annexes

FNDP Namur

Septembre 1987

Mise ONLINE de machines d'analyses biologiques dans le cadre d'un programme de gestion de laboratoire

Annexes

P. Wautié

Année académique 1986-1987

Promoteur: J.FICHEFET

VALIDATION

Texte des instructions du
programme de validation.

```
[inherit ('sys$library:starlet','vl_src:valdem','vl_src:niv0','vl_src:types'),
 check(nobounds)]
program ddffd(input,output);

label 9999;

const nbre_fonction = 12;
      nbre_erreur = 11;
      csi= chr(27);
      char_manquant = '-';
      nbre_char_manquant = 4;
      nbre_char_nom_ana = 5;
      nbre_tentative_error = 10;
      nbre_ligne_en_suspend = 40;
      deplig = 8;
      depcol = 19;
      fill_char = '-';

type

        spec_char =(UC_REC,UC_INS,UC_EFF,UC_SEL,UC_PUP,UC_PDO,
                    UC_CUP,UC_CDO,UC_CRI,UC_CLE,
                    UC_PF1,UC_PF2,UC_PF3,UC_PF4,
                    UC_NOT,UC_TAU,
                    UC_F06,UC_F07,UC_F08,UC_F09,UC_F10,
                    UC_LF,
                    UC_F11,UC_F12,UC_F13,UC_F14,
                    UC_BS,
                    UC_F15,UC_F16,
                    UC_RET,
                    UC_F17,UC_F18,UC_F19,UC_F20,
                    UC_TIL,
                    UC_KMI,UC_KC0,UC_KPE,
                    UC_KRI,
                    UC_K00,UC_K01,UC_K02,UC_K03,UC_K04,
                    UC_K05,UC_K06,UC_K07,UC_K08,UC_K09,
                    UC_ESC);

        string = record case boolean of
                true:(str:varying[5000] of char);
                false :(rien : [word] 1..5000;
                        tab : packed array [1..4998] of char);
        end;

        varying_8 = varying [8] of char;

        possible_res_douteux = (ok,anormal,premier,manquant,forrun,superflu);

        colonne = record
                res_min : real;
                res_max : real;
                res_len : integer;
                col_len : integer;
                nom : packed array [1..5] of char;
                format : packed array[1..10] of char;
                tar_col : boolean;
                num_col : integer;
                next_col : ^colonne;
                prev_col : ^colonne;
        end;

        pointeur_colonne = ^colonne;

        res_douteux = record    (* pour journal des modif de bd *)
                res_dout_ch : char;   (* + , - *)
```

```
        res_dout_ident_ana : [key(0)] packed array[1..11] of char;
        res_dout_JBL : integer;
        res_dout_res : real;
end;

res = record
        res_aff : varying[20] of char;
        res_num_col : integer;
        res_val : real;
        res_tar : boolean;
        res_pro : boolean;
        res_type :possible_res_douteux;
        modif : boolean;
        exist : boolean;
        next_res : ^res;
        prev_res : ^res;
end;

pointeur_res = ^res;

ligne = record
        ident : packed array[1..6] of char;
        num_lig : packed array[1..6] of char;
        num_godet : integer;
        next_lig : ^ligne;
        prev_lig : ^ligne;
        res_lig : ^res;
        res_dout : packed array[1..99] of boolean;
end;

pointeur_ligne = ^ligne;

commande = record
        str : varying[80] of char;
        next_com : ^commande;
        prev_com : ^commande;
end;

pointeur_commande = ^commande;

res_temp = record
        rt_jbl_lig_pos :[key(0)] packed array[1..10] of char;
        rt_res : real;
        rt_num_JBL : integer;
        rt_num_col : integer;
        rt_num_lig : integer;
        rt_num_pos : integer;
        rt_tar : boolean;
        rt_pro : boolean;
        rt_acc : boolean;
end;

curseur = record
        lig:integer;
        col:integer;
        poscol : integer;
        poslig : integer;
end;

varying_80 = varying[80] of char;
```

```
var    (* global *)

UC_tab : array [1..51] of spec_char;
com_tab : array [0..nbre_fonction] of ^commande;
err_tab : array[1..nbre_erreur] of varying_80;

curs : curseur;

buffer : ^ligne;

ptr_ligne : ^ligne;
ptr_resultat : ^res;
ptr_colonne : ^colonne;
ptr_commande : ^commande;

first_colonne : ^colonne;
first_ligne : ^ligne;

nbre_ligne_tableau : integer;
nbre_colonne_tableau : integer;

tableau_vide : boolean;

prem_lig : integer;
prem_col : integer;
der_lig : integer;
der_col : integer;

prem_lig_fe : integer;
prem_col_fe : integer;
der_lig_fe : integer;
der_col_fe : integer;

nbre_lig_fe : integer;
nbre_col_fe : integer;

dev_chan : [word] -32768..32767;

titre_jbl : varying [75] of char;

date_sys : packed array [1..6] of char; (* JJMMAA *)
heure_sys : packed array[1..4] of char; (* HHMM *)

num_JBL : integer;

partiel : boolean;

(* var program use *)
ch:char;
chspec:spec_char;
stop:boolean;
pwstat,x,y,i:integer;
ptr_lig : ^ligne;
pwreel : real;
modif,high,fin_pgm,trouve,error : boolean;
str : varying [80] of char;
```

```
procedure acces_colonne (num:integer;var ptr_col : pointeur_colonne);

var i:integer;

begin
  ptr_col := first_colonne;
  for i:=1 to num do ptr_col := ptr_col^.next_col;
end;    (* of acces_colonne *)


procedure acces_ligne (num:integer;var ptr_lig : pointeur_ligne);

var i:integer;

begin
  ptr_lig:=first_ligne;
  for i:=1 to num do ptr_lig := ptr_lig^.next_lig;
  curs.lig:=num;
end;    (* of acces_ligne *)


procedure acces_resultat (num:integer;var ptr_res : pointeur_res);

var i:integer;

begin
  for i:=1 to num do ptr_res := ptr_res^.next_res;
  curs.col:=num;
end;    (* of acces_resultat *)


procedure find_res (x,y:integer);

begin
  if (x<=nbre_ligne_tableau) and (x>0) and (y<=nbre_colonne_tableau) and (y>0)
        then begin
                acces_ligne(x,ptr_ligne);
                ptr_resultat := ptr_ligne^.res_lig;
                acces_resultat (y,ptr_resultat);
                acces_colonne (y,ptr_colonne);
             end
        else begin (* attention au tab vide *) end;
end;    (* of find_res *)
```

```
procedure creer_ptr_lig (var ptr_lig:pointeur_ligne);

var temp_lig : ^ligne;
    ptr_res : ^res;
begin
  new(temp_lig);
  temp_lig^.prev_lig := ptr_lig;
  temp_lig^.next_lig := ptr_lig^.next_lig;
  ptr_lig^.next_lig := temp_lig;
  temp_lig^.next_lig^.prev_lig := temp_lig;
  new(temp_lig^.res_lig);
(* init for first res in the line *)
  ptr_res := temp_lig^.res_lig;
  ptr_res^.next_res :=ptr_res;
  ptr_res^.prev_res := ptr_res;
  ptr_res^.res_num_col := 0;
  ptr_res^.res_type := premier;
  ptr_lig:=temp_lig;
end;    (* of creer_ptr_lig *)


procedure creer_ptr_res (var ptr_res : pointeur_res );

var temp_res : ^res;

begin
  new(temp_res);
  temp_res^.prev_res := ptr_res;
  temp_res^.next_res := ptr_res^.next_res;
  ptr_res^.next_res := temp_res;
  temp_res^.next_res^.prev_res := temp_res;
  ptr_res:=temp_res;
  ptr_res^.res_type := ok;
  ptr_res^.res_pro := true;
  ptr_res^.res_tar := true;
end;    (* of creer_ptr_res *)


procedure creer_ptr_col (var ptr_col : pointeur_colonne );

var temp_col : ^colonne;

begin
  new(temp_col);
  temp_col^.prev_col := ptr_col;
  temp_col^.next_col := ptr_col^.next_col;
  ptr_col^.next_col := temp_col;
  temp_col^.next_col^.prev_col := temp_col;
  ptr_col:=temp_col;
end;    (* of creer_ptr_col *)


procedure creer_ptr_com (var ptr_com : pointeur_commande );

var temp_com : ^commande;

begin
  new(temp_com);
  temp_com^.prev_com := ptr_com;
  temp_com^.next_com := ptr_com^.next_com;
  ptr_com^.next_com := temp_com;
  temp_com^.next_com^.prev_com := temp_com;
  ptr_com:=temp_com;
end;    (* of creer_ptr_com *)
```

```
procedure lec_touche(var touche:char; var special:spec_char);

label  9;

var x,stat : integer;
    toto : packed array [1..1] of char;

(* UC_  : unprintable char *)
begin
   touche := chr(0);
   special := UC_not;
   stat := $qiow(,dev_chan,io$_readvblk+io$m_noecho, , , ,toto,1);
   case ord(toto[1]) of
        27:begin
             stat:=$qiow(,dev_chan,io$_readvblk+io$m_noecho, , , ,toto,1);
             if not(toto[1] in [chr(27),'[','0']) then goto 9;
             if toto[1] = chr(27) then special:=UC_ESC
             else
             begin
               stat:=$qiow(,dev_chan,io$_readvblk+io$m_noecho, , , ,toto,1);
               case toto[1] of
               'P','Q','R','S' : special := UC_tab[ord(toto[1])-69];
               'A','B','C','D' : special := UC_tab[ord(toto[1])-58];
               'l','m','n','o','p','q','r','s','t',
               'u','v','w','x','y' : special := UC_tab[ord(toto[1])-71];
               'M' : special := UC_RET;
               '0','1','2','3','4','5','6','7','8','9': begin
           x:=ord(toto[1])-48;
           stat :=$qiow(,dev_chan,io$_readvblk+io$m_noecho, , , ,toto,1);
           if not (toto[1] in ['0'..'9','~']) then goto 9;
           if (toto[1] in ['0'..'9']) then begin
                x:=x*10+ord(toto[1])-48;
                stat :=$qiow(,dev_chan,io$_readvblk+io$m_noecho, , , ,toto,1);
                if toto[1] <> '~' then goto 9;
                                       end;
           if not (x in [1..6,17..19,20,21,23..26,28,29,31..34]) then goto 9;
           special := UC_tab[x];
                                     end;
                 otherwise goto 9;
               end; (* of case toto[1] *)
              end; (* of if *)
            end; (* of case 27 *)
        8  : special := UC_BS;
        9  : begin touche := toto[1];special := UC_TAU; end;
        10 : special := UC_LF;
        13 : begin special := UC_RET; touche := toto[1]; end;
  39,126,96: begin special := UC_TIL; touche := toto[1]; end;
 otherwise   touche:=toto[1];
        end; (* of case *)
9: ;
end;    (* of lec_touche *)


procedure afficher_next_com (var ptr_com : pointeur_commande);

var stat : integer;

begin
        ptr_com := ptr_com^.next_com;
        if substr(ptr_com^.str,1,10) = 'NO COMMENT' then
                                              ptr_com := ptr_com^.next_com;
        stat := $put_screen(ptr_com^.str,23,1,3);
end;    (* of afficher_next_com *)


procedure error_sys_stop;
```

```
begin
  bell;bell;bell;
  failing_open;
  goto 9999;
end;    (* of error_sys_stop *)


procedure error_sys_more;

var ch : char;
    chspec : spec_char;

begin
  bell;
  display_error(err_tab[8],1);
  lec_touche(ch,chspec);
  case ch of 'q','Q' : goto 9999;
  end; (* of case *)
  erase_error;
end;    (* of error_sys_more *)
```

```
(*      *       *       *       procedure ecran *       *       *       *)

function gitixy(x,y:integer):varying_8;

var str:varying[8] of char;
    i:integer;

begin
writev(str,csi,'[',x:1,';',y:1,'H');
gitixy := str;
end;    (* of gitixy *)


procedure make_res_affich(i,j:integer;ptr_res:pointeur_res;
                                  var str_res:varying[c] of char);
var ptr_col : ^colonne;
   num, z : integer;
   str : varying [20] of char;

begin
i:=deplig + i - prem_lig_fe;
num:=depcol;
acces_colonne(prem_col_fe,ptr_col);
for z:=prem_col_fe to (j-1) do
begin
  num := num + ptr_col^.col_len;
  ptr_col := ptr_col^.next_col;
end;
(* num := num + (ptr_col^.col_len - ptr_col^.res_len) div 2; *)
j:=num;
str := '';
str_res := gitixy(i,j) + pad('',' ',ptr_colonne^.res_len);
if ptr_res^.exist then begin
        convertir_ecrire(ptr_col^.format,ptr_res^.res_val,ptr_res^.res_tar,
                         ptr_res^.res_pro,str);
        writev(str_res,csi,'[0;',ord(ptr_res^.res_type):1,'m',str,csi,'[0m');
        str_res :=  gitixy (i,j) + str_res;
                          end;
if ptr_res^.res_type = manquant then str_res:=
                gitixy (i,j) + pad('',char_manquant,ptr_colonne^.res_len);

end;    (* of make_res_affich *)


procedure affich_resultat;

var stat : integer;
    buf : string;

begin
   make_res_affich(curs.lig,curs.col,ptr_resultat,buf.str);
   stat := $qiow( ,dev_chan,io$_writevblk, , , ,buf.tab,length(buf.str));
end;    (* of affich_resultat *)


procedure affich_fenetre;


var buf_com:string;
    i,j:integer;
    str_res : varying[30] of char;
    ptr_col : ^colonne; ptr_lig : ^ligne; ptr_res : ^res;
    stat : integer;
    temp_curs : curseur;
    pla_col : integer;
```

```
begin
  temp_curs := curs;
  buf_com.str := '';
  pla_col := depcol;
  ptr_col := ptr_colonne;
  ptr_lig := ptr_ligne;
  ptr_res := ptr_resultat;
  acces_colonne (prem_col_fe,ptr_colonne);
  buf_com.str := gitixy(deplig-2,0) + csi + '[K' + ' god        ref ';
  for i:= prem_col_fe to der_col_fe do
  begin
        writev(str_res,ptr_colonne^.nom:5);
        buf_com.str := buf_com.str + gitixy (deplig-2,pla_col +
                (ptr_colonne^.col_len - nbre_char_nom_ana ) div 2) + str_res;
        pla_col := pla_col + ptr_colonne^.col_len;
        ptr_colonne := ptr_colonne^.next_col;
  end;
  buf_com.str := buf_com.str + gitixy(deplig-1,0) + csi + '(0' +
                   pad('',chr(115),80) + csi +'(B';
  for i:=prem_lig_fe to der_lig_fe do
  begin
        acces_colonne (prem_col_fe,ptr_colonne);
        find_res(i,prem_col_fe);
        writev(str_res,ptr_ligne^.num_godet:6,' ',csi,'(0',chr(120),csi,'(B',
                ' ',ptr_ligne^.ident:6,' ',csi,'(0',chr(120),csi,'(B');
        buf_com.str := buf_com.str + gitixy(i-prem_lig_fe+deplig,0)+csi +
                        '[K' + str_res;
        for j:=prem_col_fe to der_col_fe do
        begin
                make_res_affich(i,j,ptr_resultat,str_res);
                buf_com.str := buf_com.str + str_res;
                ptr_resultat := ptr_resultat^.next_res;
                ptr_colonne := ptr_colonne^.next_col;
        end;
        j := length(buf_com.str);
        stat := $qiow( ,dev_chan,io$_writevblk, , , ,
                        buf_com.tab,length(buf_com.str));
        buf_com.str := '';
   end;
   ptr_colonne := ptr_col;
   ptr_ligne := ptr_lig;
   ptr_resultat := ptr_res;
   curs := temp_curs;

end;    (* of affich_fenetre *)


procedure calcul_pos_curs;

var i:integer;
    ptr_col : ^colonne;

begin
  acces_colonne(prem_col_fe,ptr_col);
  curs.poscol := depcol;
  for i:= prem_col_fe to (curs.col - 1) do
  begin
    curs.poscol := curs.poscol + ptr_col^.col_len;
    ptr_col := ptr_col^.next_col;
  end;
end;    (* of calcul_pos_curs *)


procedure sup_pos_curs;

var buf_com : string;
```

```
    str : varying[20] of char;

begin
   str := gitixy(curs.lig - prem_lig_fe + deplig,curs.poscol - 1);
   buf_com.str := str + ' ' + str;
   stat := $qiow( ,dev_chan,io$_writevblk, , , ,
                        buf_com.tab,length(buf_com.str));
end;    (* of sup_pos_curs *)


procedure set_pos_curs;

var buf_com : string;
    stat : integer;

begin
   buf_com.str := ' ' + gitixy (curs.lig - prem_lig_fe + deplig,curs.poscol -
                  + '>' + chr(8);
   stat := $qiow( ,dev_chan,io$_writevblk, , , ,
                        buf_com.tab,length(buf_com.str));
end;    (* of position curseur *)

[global] procedure calcul_fe_pt(x,y:integer); forward;
                (* normalement curs.lig,curs.col *)
```

```
        while (i>= prem_col) and (long<=nbre_col_fe) do
        begin
                long := ptr_col^.col_len + long;
                i := i-1;
                ptr_col := ptr_col^.prev_col;
        end;
        ptr_col := ptr_temp;
        prem_col_fe := i + 1 + ord(long>nbre_col_fe);
        sup_i := (long > nbre_col_fe);
        i := j;
        while (i<=nbre_colonne_tableau) and (long<=nbre_col_fe) do
        begin
                long := ptr_col^.col_len + long;
                i := i + 1;
                ptr_col := ptr_col^.next_col;
        end;
        der_col_fe := i - 1 - ord(long>nbre_col_fe) + ord(sup_i);
        if curs.col > der_col_fe then curs.col := der_col_fe;
        calcul_pos_curs;
        find_res(curs.lig,curs.col);
        if (old_lig <> prem_lig_fe) or (old_col <> der_col_fe)
                then affich_fenetre
                else display_error(err_tab[3],1);
      end;
UC_LF,UC_F13,UC_PUP:begin
        der_lig_fe := der_lig_fe + nbre_lig_fe;
        if der_lig_fe > der_lig then der_lig_fe := der_lig;
        prem_lig_fe := der_lig_fe - nbre_lig_fe + 1;
        if prem_lig_fe < prem_lig then prem_lig_fe := prem_lig;
        if curs.lig < prem_lig_fe then curs.lig := prem_lig_fe;
        find_res(curs.lig,curs.col);
        if (old_lig <> prem_lig_fe) or (old_col <> der_col_fe)
                then affich_fenetre
                else display_error(err_tab[2],1);
       end;
UC_PF3,UC_PD0:begin
        acces_colonne(der_col_fe,ptr_col);
        ptr_temp := ptr_col;
        i := der_col_fe + 1;
        j := der_col_fe;
        long :=0;
        ptr_col := ptr_col^.next_col;
        while (i<=nbre_colonne_tableau) and (long<=nbre_col_fe) do
        begin
                long := ptr_col^.col_len + long;
                i := i + 1;
                ptr_col := ptr_col^.next_col;
        end;
        ptr_col := ptr_temp;
        der_col_fe := i - 1 - ord(long>nbre_col_fe);
        sup_i := (long > nbre_col_fe);
        i := j;
        while (i>= prem_col) and (long<=nbre_col_fe) do
        begin
                long := ptr_col^.col_len + long;
                i := i-1;
                ptr_col := ptr_col^.prev_col;
        end;
        prem_col_fe := i + 1 + ord(long>nbre_col_fe) - ord(sup_i);
        if curs.col < prem_col_fe then curs.col := prem_col_fe;
        calcul_pos_curs;
        find_res(curs.lig,curs.col);
        if (old_lig <> prem_lig_fe) or (old_col <> der_col_fe)
                then affich_fenetre
                else display_error(err_tab[4],1);
      end;
```

```
var i,j,long : integer;
    ptr_temp,ptr_col : ^colonne;
    sup_i : boolean;
    old_lig, old_col : integer;

begin
 old_lig := prem_lig_fe; old_col := der_col_fe;
 if (not high_deplac) then begin
        case special of
        (* for deplacement curseur *)
        UC_CUP:if curs.lig > prem_lig then begin
                        curs.lig:=curs.lig-1;
                        ptr_ligne := ptr_ligne^.prev_lig;
                        ptr_resultat := ptr_ligne^.res_lig;
                        acces_resultat(curs.col,ptr_resultat);
                        if curs.lig <prem_lig_fe then deplacement(UC_INS,false)
                                                 end
                                else display_error(err_tab[1],1);
        UC_CDO:if curs.lig < der_lig then begin
                        curs.lig:=curs.lig+1;
                        ptr_ligne := ptr_ligne^.next_lig;
                        ptr_resultat := ptr_ligne^.res_lig;
                        acces_resultat(curs.col,ptr_resultat);
                        if curs.lig >der_lig_fe then deplacement(UC_PUP,false);
                                                 end
                                else display_error(err_tab[2],1);

        UC_CLE:if curs.col > prem_col then begin
                        curs.col := curs.col - 1;
                        ptr_colonne := ptr_colonne^.prev_col;
                        curs.poscol := curs.poscol - ptr_colonne^.col_len;
                        ptr_resultat := ptr_resultat^.prev_res;
                        if curs.col< prem_col_fe then deplacement(UC_SEL,false)
                                                 end
                                else display_error(err_tab[3],1);

        UC_CRI:if curs.col < der_col then begin
                        curs.col := curs.col + 1;
                        curs.poscol := curs.poscol + ptr_colonne^.col_len;
                        ptr_resultat := ptr_resultat^.next_res;
                        ptr_colonne := ptr_colonne^.next_col;
                        if curs.col > der_col_fe then deplacement(UC_PDO,false)
                                                 end
                                else display_error(err_tab[4],1);

        (* for deplacement fenetre *)
        UC_BS,UC_F12,UC_INS:begin
                prem_lig_fe := prem_lig_fe - nbre_lig_fe;
                if prem_lig_fe < prem_lig then prem_lig_fe := prem_lig;
                der_lig_fe := prem_lig_fe + nbre_lig_fe - 1;
                if der_lig_fe > der_lig then der_lig_fe := der_lig;
                if curs.lig > der_lig_fe then curs.lig := der_lig_fe;
                find_res(curs.lig,curs.col);
                if (old_lig <> prem_lig_fe) or (old_col <> der_col_fe)
                        then affich_fenetre
                        else display_error(err_tab[1],1);
              end;
        UC_PF2,UC_SEL:begin
                acces_colonne(prem_col_fe,ptr_col);
                ptr_temp := ptr_col;
                i := prem_col_fe - 1;
                j := prem_col_fe;
                long :=0;
                ptr_col := ptr_col^.prev_col;
```

```
        end; (* of case *)
                    end  (* of then *)
                  else begin
        case special of
        (* for high deplacement curseur *)
        UC_CUP:curs.lig := prem_lig_fe;
        UC_CDO:curs.lig := der_lig_fe;
        UC_CLE:curs.col := prem_col_fe;
        UC_CRI:curs.col := der_col_fe;

        (* for high deplacement fenetre *)
        UC_F12,UC_BS,UC_INS:curs.lig := prem_lig;
        UC_PF2,UC_SEL:curs.col := prem_col;
        UC_F13,UC_LF,UC_PUP:curs.lig := der_lig;
        UC_PF3,UC_PDO:curs.col := der_col;
        end; (* of case *)
        find_res (curs.lig,curs.col);
        calcul_pos_curs;
        if (curs.lig > der_lig_fe) or (curs.lig < prem_lig_fe) or
             (curs.col > der_col_fe) or (curs.col < prem_col_fe) then
                calcul_fe_pt(curs.lig,curs.col);

                    end;  (* of else *)

end;    (* of deplacement *)
```

```
procedure calcul_fe_pt (* (x,y:integer) *);  (* normalement curs.lig,curs.col

var long : integer;
    ptr_col : ^colonne;

begin
  prem_lig_fe := x - (nbre_lig_fe div 2);
  if prem_lig_fe < prem_lig then prem_lig_fe := prem_lig;
  der_lig_fe := prem_lig_fe + nbre_lig_fe - 1;
  if der_lig_fe > der_lig then der_lig_fe := der_lig;
  prem_lig_fe := der_lig_fe - nbre_lig_fe + 1;
  if prem_lig_fe < prem_lig then prem_lig_fe := prem_lig;

  acces_colonne( y , ptr_col);
  long := - (ptr_col^.col_len div 2);
  while ( y >= prem_col ) and (long < nbre_col_fe div 2) do
  begin
        long := long + ptr_col^.col_len;
        ptr_col := ptr_col^.prev_col;
        y := y - 1;
  end;
  der_col_fe := y + ord(long > (nbre_col_fe) div 2);
  deplacement (UC_PDO,false);
end;    (* of calcul_fe_pt *)
```

```
(*      *       *       *       procedure ???   *       *       *       *)

procedure instring (var str:varying[c] of char;num,lig,col,mode:integer;
                        fill_char:char;x:integer;var modif:boolean;y:integer);

var sais : integer;
    fin : boolean;

begin
 fin := (mode<=0);
 if (length(str)>mode) and (mode<>0) then str := substr(str,1,mode);
 while not fin do
 begin
   modif := true;
   sais := saisie_edition(str,num,lig,col,mode,fill_char,x,modif,y);
   fin := (sais = 10);
   if sais = -10 then afficher_next_com(ptr_commande);
 end;
end;    (* of instring *)
```

```
(*      *       *       * procedure travail sur un res *       *       *)

procedure modifier_res (ptr_res : pointeur_res; ptr_lig : pointeur_ligne ;
                                tar,pro : boolean; resultat : real);

var ptr_col : ^colonne;
    vrais : boolean;

begin
  ptr_res^.res_tar := tar;
  ptr_res^.res_pro := pro;
  ptr_res^.exist := true;
  ptr_res^.modif := true;
  ptr_res^.res_val := resultat;
(* verification demande effectuee *)
  if ptr_lig^.res_dout[curs.col] = false then ptr_res^.res_type := superflu
                                         else ptr_res^.res_type := ok;
(* verification resultat dans les normes *)
  acces_colonne(curs.col,ptr_col);
  valider_vraissembl (resultat,ptr_col^.res_min,ptr_col^.res_max,vrais);
  if not vrais then ptr_res^.res_type := anormal;
end; (* of modifier_res *)

procedure supprimer_res (ptr_res : pointeur_res; ptr_lig : pointeur_ligne );

begin
  buffer^.res_lig^.res_val := ptr_res^.res_val;   (* pour CB res *)
  ptr_res^.res_val := 0;
  ptr_res^.exist := false;
  ptr_res^.modif := true;
  if (ptr_lig^.res_dout[curs.col] = true) then ptr_res^.res_type := manquant
                                          else ptr_res^.res_type := ok;
end;    (* of supprimer_res *)

procedure mod_add_sup_res (ch:char;num:integer;ptr_lig:pointeur_ligne;
                           ptr_res:pointeur_res;var stop :boolean);

var stat:integer;
    realstr : real;
    str,str_error:varying[80] of char;
    modif,correct,res_term,normal : boolean;
    lc : integer;

procedure affich_format(ptr_col : pointeur_colonne);
var stat : integer;
    str : varying[10] of char;
begin
  case ptr_col^.format[1] of
  'D','I':str := pad('','*',ord(ptr_col^.format[2])-48);
  'F'    :str := '1/' + pad('','*',ord(ptr_col^.format[2])-48);
  'B'    : str:= '***';
  'R'    : str := pad('','*',ord(ptr_col^.format[2])-48) + '.' +
                  pad('','*',ord(ptr_col^.format[4])-48);
  end; (* of case *)
  stat := $put_screen(str,23,71,3);
end;    (* of affich_format *)

begin
  ptr_commande := com_tab [9];
  afficher_next_com(ptr_commande);
  str_error := '';
  set_pos_curs;
  acces_colonne(curs.col,ptr_colonne);
  affich_format(ptr_colonne);
  if ptr_res^.exist then convertir_ecrire (ptr_colonne^.format,
                                           ptr_res^.res_val,ptr_res^.res_tar,
```

```
                                                    ptr_res^.res_pro,str)
                    else str:='';
  correct := false;
  while not correct do
    begin
        display_error(str_error,1);
        correct := true;
        instring(str,num + 1,curs.lig - prem_lig_fe + deplig,curs.poscol,
                 ptr_colonne^.res_len,fill_char,0,modif,0);
        if (length(str)<>0) then begin
                case str[1] of '.': begin stop:=true;
                                          correct:=true;
                                    end;

                               '\': begin
                                        supprimer_res(ptr_res,ptr_ligne);
                                        correct := true;
                                    end;

                               otherwise
                                    begin
                                        valider_format(ptr_colonne^.format,str
                                        realstr,ptr_res^.res_tar,
                                        ptr_res^.res_pro,
                                        res_term,correct,normal,str_error);
                                        modifier_res(ptr_res,ptr_ligne,
                                        ptr_res^.res_tar,ptr_res^.res_pro,
                                        realstr);
                                        if ptr_res^.res_type = anormal
                                              then ptr_res^.res_type := ok;
                                     end;
                               end; (* case *)
                        end (* then *)
                        else    begin
                                        supprimer_res(ptr_res,ptr_ligne);
                                        correct := true;
                               end;
     end; (* of while *)
    affich_resultat;
    stat := $put_screen('           ',23,71,1);
    ptr_commande := com_tab[1];
    afficher_next_com(ptr_commande);

end;    (* of mod_add_sup_res *)
```

```
procedure modif_bd (ch:char); (* + , - *)

var trans : res_douteux;
    str : varying [15] of char;


procedure sauv_res_dout ( trans : res_douteux);

var fich_jour : file of res_douteux;

begin
   open(fich_jour,'vl_online:journal_online',unknown, organization := indexed,
          access_method := keyed , sharing := readwrite,error := continue);
   resetk(fich_jour,0,error := continue);
   findk(fich_jour,0,trans.res_dout_ident_ana,error := continue);
   fich_jour^ := trans;
   if ufb(fich_jour) then put(fich_jour,error := continue)
                     else update (fich_jour,error := continue);
   close(fich_jour,error := continue);
end;    (* of sauv_res_dout *)

begin
   trans.res_dout_ch := ch;
   writev(str,ptr_ligne^.ident,ptr_colonne^.nom);
   readv(str,trans.res_dout_ident_ana);
   trans.res_dout_JBL := num_JBL;
   trans.res_dout_res := ptr_resultat^.res_val;
   sauv_res_dout (trans);
end;    (* of modif_bd *)
```

```
(*      *       *       *  procedure niveau superieures *       *       *)

procedure find_resultat (valeur:real);

var num_lig : integer;
    num_col : integer;
    trouve : boolean;

begin
  trouve := false;
  num_lig := curs.lig;
  num_col := curs.col;
  while (num_lig <= nbre_ligne_tableau) and (not trouve) do
  begin
        while (num_col <= nbre_colonne_tableau) and (not trouve) do
        begin
                trouve := (ptr_resultat^.res_val = valeur ) and
                                (ptr_resultat^.exist);
                num_col := num_col + 1;
                ptr_resultat := ptr_resultat^.next_res;
        end;
        if (not trouve) then num_col := prem_col;
        num_lig := num_lig + 1;
        ptr_ligne := ptr_ligne^.next_lig;
        ptr_resultat := ptr_ligne^.res_lig^.next_res;
  end;
  if trouve then begin curs.lig := num_lig-1;curs.col := num_col-1;end
            else display_error(err_tab[5],1);
  if (curs.lig > der_lig_fe) or (curs.lig < prem_lig_fe) or
     (curs.col > der_col_fe) or (curs.col < prem_col_fe) then
                calcul_fe_pt(curs.lig,curs.col);
  find_res(curs.lig,curs.col);
  calcul_pos_curs;
end;    (* of find_resultat *)
```

```
procedure intro_auto ;

var num_lig : integer;
    num_col : integer;
    stop : boolean;

begin
  ptr_commande := com_tab[2];
  afficher_next_com (ptr_commande);
  num_lig := curs.lig;
  acces_ligne(curs.lig,ptr_ligne);
  ptr_ligne := ptr_ligne^.prev_lig;
  stop := false;
  sup_pos_curs;
  while (num_lig<=nbre_ligne_tableau) and (not stop) do
  begin
        ptr_ligne := ptr_ligne^.next_lig;
        num_col := prem_col;
        ptr_resultat := ptr_ligne^.res_lig;
        curs.lig := num_lig;
        while (num_col <= nbre_colonne_tableau) and (not stop) do
        begin
                curs.col := num_col;
                ptr_resultat := ptr_resultat^.next_res;
                if ptr_resultat^.res_type = manquant then begin
                   if (curs.lig > der_lig_fe) or (curs.lig < prem_lig_fe) or
                         (curs.col > der_col_fe) or (curs.col < prem_col_fe)
                               then calcul_fe_pt(curs.lig,curs.col);
                               calcul_pos_curs;
                               mod_add_sup_res ('-',1,ptr_ligne,
                                                ptr_resultat,stop);
                               sup_pos_curs;
                end; (* IF *)
                num_col := num_col + 1;
        end;
        num_lig := num_lig + 1;
   end;
   find_res(curs.lig,curs.col);
   calcul_pos_curs;
   if (curs.lig > der_lig_fe) or (curs.lig < prem_lig_fe) or
      (curs.col > der_col_fe) or(curs.col < prem_col_fe)
     then calcul_fe_pt(curs.lig,curs.col);
   display_error(err_tab[9],1);
   ptr_commande := com_tab[1];
   afficher_next_com (ptr_commande);
end;    (* of intro_auto *)
```

```
procedure correct_auto ;

const return = chr (13);
      space = chr (32);
      tab = chr (9);
      backslash = '\';
      nul = chr(0);

var num_lig : integer;
    num_col : integer;
    stop : boolean;
    touche : char;
    special : spec_char;

begin
  ptr_commande := com_tab[3];
  afficher_next_com (ptr_commande);
  sup_pos_curs;
  num_lig := curs.lig;
  acces_ligne (curs.lig,ptr_ligne);
  ptr_ligne := ptr_ligne^.prev_lig;
  stop := false;
  while (num_lig<=nbre_ligne_tableau) and (not stop) do
  begin
        ptr_ligne := ptr_ligne^.next_lig;
        num_col := prem_col;
        ptr_resultat := ptr_ligne^.res_lig;
        curs.lig := num_lig;
        while (num_col <= nbre_colonne_tableau) and (not stop) do
        begin
                ptr_resultat := ptr_resultat^.next_res;
                if ptr_resultat^.res_type <> ok  then
                        begin
                                curs.col := num_col;
                                if (curs.lig > der_lig_fe) or
                                     (curs.lig < prem_lig_fe) or
                                       (curs.col > der_col_fe) or
                                         (curs.col < prem_col_fe) then
                                          calcul_fe_pt(curs.lig,curs.col);
                                calcul_pos_curs;
                                set_pos_curs;
                                find_res(curs.lig,curs.col);
                                lec_touche (touche,special);
                                case touche of
                                return :begin
                                          ptr_resultat^.modif := true;
                        if ptr_resultat^.res_type=anormal then
                                if ptr_ligne^.res_dout[curs.col] = true then
                                          ptr_resultat^.res_type := ok
                                                                  else
                                          ptr_resultat^.res_type := superflu;
                         if ptr_resultat^.res_type=manquant
                                then begin modif_bd('-');
                                ptr_ligne^.res_dout[curs.col] := false;
                                        ptr_resultat^.res_type := ok;
                                end;
                          if ptr_resultat^.res_type=superflu
                                then begin modif_bd('+');
                                ptr_ligne^.res_dout[curs.col] := true;
                                ptr_resultat^.res_type := ok;
                                     end;
                                          end;
                                space  : ;
                             backslash : supprimer_res(ptr_resultat,ptr_ligne);
                                tab    : begin
                                           num_col := num_col - 1;
```

```
                                                    ptr_resultat^.prev_res;
                                                  ptr_colonne := ptr_colonne^.prev_co
                                                  afficher_next_com(ptr_commande);
                                                end;
                                      nul     : if special = UC_PF1 then stop:=true;
                                      otherwise mod_add_sup_res (touche,1,ptr_ligne,
                                                 ptr_resultat,stop);
                                      end;  (* of case *)
                                      affich_resultat;
                                      sup_pos_curs;
                            end;
                    num_col := num_col + 1;
            end;
            num_lig := num_lig + 1;
    end;
    find_res(curs.lig,curs.col);
    calcul_pos_curs;
    if (curs.lig > der_lig_fe) or (curs.lig < prem_lig_fe) or
       (curs.col > der_col_fe) or(curs.col < prem_col_fe)
     then calcul_fe_pt(curs.lig,curs.col);
    display_error(err_tab[10],1);
    ptr_commande := com_tab[1];
    afficher_next_com (ptr_commande);
end;    (* of correct_auto *)
```

```
(*      *       *       * procedure sur ligne de res *  *       *       *)

procedure supprimer_ligne_res (var ptr_lig : pointeur_ligne);

var ptr : ^res;
    temp : ^res;

begin
  ptr := buffer^.res_lig^.next_res;
  while ptr^.res_type <> premier do
  begin
        temp := ptr^.next_res;
        dispose (ptr);
        ptr := temp;
  end;
  dispose (ptr);
  buffer^.res_lig := ptr_lig^.res_lig;
end;    (* of supprimer_ligne_res *)


procedure ajouter_ligne_res ( var ptr_lig : pointeur_ligne );

var ptr : ^res;
    i : integer;
    temp : integer;

begin
  temp := curs.col;
  new(ptr);
  ptr^.next_res := ptr;
  ptr^.prev_res := ptr;
  ptr^.res_type := premier;
  ptr_lig^.res_lig := ptr;
  for i:=1 to nbre_colonne_tableau do begin creer_ptr_res (ptr);
                                            curs.col := i;
                                            supprimer_res(ptr,ptr_lig);
                                      end;
  curs.col := temp;
end;    (* of ajouter_ligne_res *)


procedure copy_buffer(ptr_lig : pointeur_ligne);

var temp:^ligne;
    i:integer;
    ptr_res,ptr_res_buffer : ^res;
    temp_col : integer;

begin
  temp := buffer;
  temp_col := curs.col;
  buffer := ptr_lig;
  supprimer_ligne_res(ptr_lig);
  ajouter_ligne_res(ptr_lig);
  buffer := temp;
  ptr_res := ptr_lig^.res_lig^.next_res;
  ptr_res_buffer := buffer^.res_lig^.next_res;
  for i:=1 to nbre_colonne_tableau do
  begin
        curs.col:=i;
        if ptr_res_buffer^.exist
          then modifier_res(ptr_res,ptr_lig,ptr_res_buffer^.res_tar,
                ptr_res_buffer^.res_pro,ptr_res_buffer^.res_val)
          else supprimer_res (ptr_res,ptr_lig);
        ptr_res := ptr_res^.next_res;
        ptr_res_buffer := ptr_res_buffer^.next_res;
```

```
  curs.col := temp_col;
  find_res(curs.lig,curs.col);
  affich_fenetre;
end;    (* of copy_buffer *)


procedure add_lig_res;

var ptr_lig : ^ligne;
    num_lig : integer;
    i,x     : integer;
    ptr_res : ^res;

begin
  x:=curs.col;
  ptr_lig := first_ligne^.prev_lig;
  supprimer_ligne_res (ptr_lig);
  num_lig := der_lig;
  while num_lig > curs.lig do
  begin
       ptr_lig^.num_godet := ptr_lig^.prev_lig^.num_godet;
       ptr_lig^.res_lig := ptr_lig^.prev_lig^.res_lig;
       ptr_res :=ptr_lig^.res_lig^.next_res;
       for i:= prem_col to der_col do
       begin
         curs.col := i;
         if ptr_res^.exist then modifier_res(ptr_res,ptr_lig,ptr_res^.res_tar
                               ,ptr_res^.res_pro,ptr_res^.res_val)
                           else supprimer_res(ptr_res,ptr_lig);
         ptr_res := ptr_res^.next_res;
       end;
       ptr_lig := ptr_lig^.prev_lig;
       num_lig := num_lig - 1;
  end;
  ajouter_ligne_res (ptr_lig);
  ptr_lig^.num_godet := 0;
  curs.col := x;
end;    (* of add_lig_res *)


procedure sup_lig_res;

var ptr_lig : ^ligne;
    num_lig : integer;
    i,x     : integer;
    ptr_res : ^res;

begin
  x:=curs.col;
  acces_ligne(curs.lig,ptr_lig);
  supprimer_ligne_res (ptr_lig);
  num_lig := curs.lig;
  while num_lig < nbre_ligne_tableau do
  begin
       ptr_lig^.num_godet := ptr_lig^.next_lig^.num_godet;
       ptr_lig^.res_lig := ptr_lig^.next_lig^.res_lig;
       ptr_res :=ptr_lig^.res_lig^.next_res;
       for i:= prem_col to der_col do
       begin
         curs.col := i;
         if ptr_res^.exist then modifier_res(ptr_res,ptr_lig,ptr_res^.res_tar
                                             ptr_res^.res_pro,
                                                   ptr_res^.res_val)
                           else supprimer_res(ptr_res,ptr_lig);
         ptr_res := ptr_res^.next_res;
```

```
        ptr_lig := ptr_lig^.next_lig;
        num_lig := num_lig + 1;
  end;
  ajouter_ligne_res (ptr_lig);
  ptr_lig^.num_godet :=0;
  curs.col := x;
end;    (* of sup_lig_res *)


procedure add_sup_lig_res;

var ch:char;
    chspec : spec_char;

begin
 ptr_commande := com_tab[4];
 afficher_next_com (ptr_commande);
 lec_touche(ch,chspec);
 case ch of '+' : add_lig_res;
            '-' : sup_lig_res;
        end;
 find_res(curs.lig,curs.col);
 affich_fenetre;
end;    (* of add_sup_lig_res *)
```

```
procedure rech_ident;

var str:varying[10] of char;
    tab : packed array[1..6] of char;
    modif,noerror : boolean;
    i : integer;

function valider_int (str : varying[c] of char ; var val : integer) : boolean ;
(* int >= 0 *)

var i : integer;

begin
 val := -1;
 readv(str,val,error:=continue);
 valider_int := (val <> -1);
end;    (* of valider_int *)

begin
  str := '';
  ptr_commande := com_tab[6];
  afficher_next_com (ptr_commande);
  sup_pos_curs;
  instring(str,1,23,70,6,'*',1,modif,0);
  noerror := valider_int ( str , i );
  if noerror then
  begin
    str:=pad('','0',6-length(str)) + str;
    for i := 1 to 6 do tab[i] := str [i];
    i := prem_lig;
    trouve := false;
    ptr_lig := first_ligne;
    while (i<=nbre_ligne_tableau) and (not trouve) do
    begin
     ptr_lig := ptr_lig^.next_lig;
     trouve := (ptr_lig^.ident = tab);
     i:=i+1;
    end;
    if trouve then
                begin find_res(i-1,prem_col);
                      curs.lig := i-1;
                      curs.col := prem_col;
                end
              else display_error(err_tab[6],1);
  end
  else display_error(err_tab[11],1);
  calcul_fe_pt (curs.lig,curs.col);
  calcul_pos_curs;
  ptr_commande := com_tab[1];
  afficher_next_com (ptr_commande);
  i := $put_screen('          ',23,70,1);
end;    (* of rech_ident *)
```

```
procedure rech_val;

var noerror,modif : boolean;
    i,x,y : integer;
    str : varying[10] of char;

function valider_reel ( str : varying[c] of char; var val:real) : boolean ;
(* reel positif : ? tester status *)
var i : integer;

begin
 val := -1;
 readv(str,val,error:=continue);
 valider_reel := (val <> -1);
end;    (* of valider_reel *)

begin
  ptr_commande := com_tab[5];
  afficher_next_com (ptr_commande);
  sup_pos_curs;
  str := '';
  instring(str,1,23,70,8,'*',1,modif,0);
  noerror := valider_reel ( str ,pwreel );
  if noerror then  find_resultat(pwreel)
             else display_error(err_tab[11],1);
  ptr_commande := com_tab[1];
  afficher_next_com (ptr_commande);
  i := $put_screen('          ',23,70,1);
end;    (* of rech_val *)
```

```
var error,modif : boolean;
    i,x,y : integer;
    str : varying[10] of char;

function valider_int ( str : varying[c] of char; var val : integer) : boolean ;

var i : integer;

begin
 readv(str,val,error:=continue);
 valider_int := (val <> -1);;
end;    (* of valider_int *)

begin
  x:= curs.lig; y:=curs.col;
  str := '';
  sup_pos_curs;
  ptr_commande := com_tab[7];
  afficher_next_com (ptr_commande);
  instring(str,1,23,70,6,'*',1,modif,0);
  error := valider_int ( str , x );
  str := '';
  afficher_next_com (ptr_commande);
  instring(str,1,23,70,6,'*',1,modif,0);
  error := valider_int ( str , y);
  if x> der_lig then x:=der_lig;
  if x< prem_lig then x := prem_lig;
  if y> der_col then y:=der_col;
  if y< prem_col then y := prem_col;
  curs.lig :=x; curs.col := y;
  find_res(curs.lig,curs.col);
  calcul_fe_pt(curs.lig,curs.col);
  calcul_pos_curs;
  ptr_commande := com_tab[1];
  afficher_next_com (ptr_commande);
  i := $put_screen('          ',23,70,1);
end;    (* of jump *)
```

```
(*      *       *       * procedure initialisation *    *       *       *)

procedure lecture_num_jbl (var num_jbl : integer);

type    online = record
                 num : integer;
                 titre : varying[75] of char;
                 prev_onl : ^online;
                 next_onl : ^online;
         end;

         pointeur_online = ^online;

var first_online : ^online;
    nbre_online : integer;
    ptr_onl : ^online;
    num : integer;

procedure creer_ptr_onl (var ptr_onl : pointeur_online);

var temp:^online;

begin
  new(temp);
  temp^.next_onl := ptr_onl^.next_onl;
  temp^.prev_onl := ptr_onl;
  ptr_onl^.next_onl := temp;
  temp^.next_onl^.prev_onl := temp;
  ptr_onl := temp;
end;    (* of creer_ptr_online *)

procedure lecture_fich_param;

var inf : text;
    ptr_onl :^online;
    x,y : integer;
    res,str : varying[80] of char;
    ident : packed array[1..2] of char;
    fich_JBL : file of rec_sig_ent_jbl;

begin
    ptr_onl := first_online;
    nbre_online := 0;
    open(inf,'vl_online:param_online',readonly,error:=continue);
    open(fich_jbl,'vl_dat:sigjbl.dat',readonly,organization := indexed,
         access_method := keyed, sharing := readonly,error := continue);
    if (status(fich_jbl) <> 0) or (status(inf)<>0) then error_sys_stop;
    resetk(fich_jbl,0,error := continue);
    reset(inf,error:=continue);
    if (status(fich_jbl) <> 0) or (status(inf)<>0) then error_sys_stop;
    while status(inf) = 0 do
    begin
        readln(inf,str,error:=continue);
        readv(str,ident,x,y,res,error:=continue);
        if res[1] = '*' then
        begin
                nbre_online := nbre_online + 1;
                creer_ptr_onl(ptr_onl);
                ptr_onl^.num := y;
                findk(fich_jbl,0,y,error:=continue);
                if ufb(fich_jbl) then ptr_onl^.titre := '?????'
                                 else ptr_onl^.titre :=
                                                  fich_jbl^.sig_tit_ent_jbl;
        end;
    end;
    close(inf,error:=continue);
```

```
    close(fich_jbl,error:=continue);
end;    (* lecture_fich_par *)


procedure choix_num_JBL;

var ch : char;
    spec : spec_char;
    stat : integer;
    str : varying[80] of char;

begin
  ptr_onl := first_online^.next_onl;
  num := 1;
  spec := UC_NOT;
  repeat
        case spec of
        UC_CDO:if num > 1 then begin
                                ptr_onl := ptr_onl^.prev_onl;
                                num := num - 1;
                               end;
        UC_CUP:if num < nbre_online then begin
                                          ptr_onl := ptr_onl^.next_onl;
                                          num := num + 1;
                                         end;
        end; (* of case *)
        writev(str,ptr_onl^.num,' ',ptr_onl^.titre);
        str := str + pad('',' ',80-length(str));
        stat := $put_screen(str,10,1,3);
        lec_touche(ch,spec);
  until spec = UC_RET;
  num_JBL := ptr_onl^.num;
end;    (* of choix_num_JBL)

(* lecture_num_jbl *)

begin
  ptr_commande := com_tab[12];
  afficher_next_com (ptr_commande);

  new(first_online);

  first_online^.next_onl := first_online;
  first_online^.prev_onl := first_online;
  first_online^.num := 0;
  first_online^.titre := ' no online ';

  lecture_fich_param;
  choix_num_jbl;

  ptr_commande := com_tab[8];
  afficher_next_com(ptr_commande);

end;    (* of lecture_num_jbl *)

procedure lecture_fich_err;

var i : integer;
    fich_err : text;

begin
   for i:=1 to nbre_erreur do err_tab[i] := 'undefined error';
   open (fich_err,'vl_online:erreur.dta',readonly,error := continue);
   reset(fich_err,error := continue);
   i:=1;
   while (i<=nbre_erreur) and (status(fich err)=0) do
```

```
        readln(fich_err,err_tab[i]);
        i := i + 1;
    end;
    close (fich_err,error := continue);
end;    (* of lecture_fich_err *)


procedure lecture_fich_com;

const long = 70;

var fich_com : text;
        i:integer;
        str:varying[long] of char;

begin
  for i:=0 to nbre_fonction do
  begin
        new(com_tab[i]);
        com_tab[i]^.str := pad('NO COMMENT',' ',long);
        com_tab[i]^.next_com := com_tab[i];
        com_tab[i]^.prev_com := com_tab[i];
  end;
  open(fich_com,'vl_online:commande.dta',readonly,sharing:=readonly,
                                error := continue);
  reset(fich_com,error := continue);
  i:=1;
  while (status(fich_com)=0) and (i<=nbre_fonction) do
  begin
        readln(fich_com,str,error := continue);
        ptr_commande := com_tab[i];
        while (str<>'end') and (str<>'END') and (not eof(fich_com)) do
        begin
                creer_ptr_com(ptr_commande);
                ptr_commande^.str:=pad(str,' ',long);
                readln(fich_com,str,error := continue);
        end;
        i:=i+1;
  end;
  close (fich_com,error := continue);
end;    (* of lecture_fich_com *)
```

```
procedure lecture_colonne (num_jbl : integer);

var fich_rec_sig_ana : file of rec_sig_ana;
    fich_rec_sig_ent_jbl : file of rec_sig_ent_jbl;
    fich_rec_sig_ana_jbl : file of rec_sig_ana_jbl;
    buf,temp : packed array[1..4] of char;
    i : integer;
    ptr_col : pointeur_colonne;

begin
   open(fich_rec_sig_ent_jbl,'vl_dat:sigjbl.dat',readonly,
        organization := indexed,access_method := keyed , sharing := readonly,
        error := continue);
   if status(fich_rec_sig_ent_jbl) <> 0 then error_sys_stop;
   resetk(fich_rec_sig_ent_jbl,0,error := continue);
   if status(fich_rec_sig_ent_jbl) <> 0 then error_sys_stop;
   findk(fich_rec_sig_ent_jbl,0,num_jbl,error := continue);
   if status(fich_rec_sig_ent_jbl) <> 0 then error_sys_stop;
   titre_jbl := fich_rec_sig_ent_jbl^.sig_tit_ent_jbl;
   nbre_colonne_tableau := fich_rec_sig_ent_jbl^.sig_nb_ana_ent_jbl;
   close (fich_rec_sig_ent_jbl,error := continue);
   open(fich_rec_sig_ana_jbl,'vl_dat:siganajbl.dat',
         readonly,organization := indexed,access_method := keyed ,
         sharing := readonly,error := continue);
   if status(fich_rec_sig_ana_jbl) <> 0 then error_sys_stop;
   open(fich_rec_sig_ana,'vl_dat:sigana.dat',readonly,
        organization := indexed,access_method := keyed , sharing := readonly,
        error := continue);
   if status(fich_rec_sig_ana) <> 0 then error_sys_stop;
   resetk(fich_rec_sig_ana_jbl,0,error := continue);
   if status(fich_rec_sig_ana_jbl) <> 0 then error_sys_stop;
   resetk(fich_rec_sig_ana,0,error := continue);
   if status(fich_rec_sig_ana) <> 0 then error_sys_stop;
   ptr_col := first_colonne;
   buf[1] := chr (num_jbl div 10 + 48);
   buf[2] := chr (num_jbl mod 10 + 48);
   for i:=1 to nbre_colonne_tableau do
   begin
        buf[3] := chr (i div 10 + 48);
        buf[4] := chr (i mod 10 + 48);
        creer_ptr_col( ptr_col);
        findk (fich_rec_sig_ana_jbl,0,buf,error := continue);
        if status(fich_rec_sig_ana_jbl) <> 0 then error_sys_stop;
          ptr_col^.nom := fich_rec_sig_ana_jbl^.sig_code_ana_jbl;
          ptr_col^.col_len := fich_rec_sig_ana_jbl^.sig_larg_ecr_ana_jbl + 1;
          ptr_col^.num_col := i;

        findk (fich_rec_sig_ana,0,ptr_col^.nom,error := continue);
        if status(fich_rec_sig_ana) <> 0 then error_sys_stop;
          ptr_col^.res_min := fich_rec_sig_ana^.sig_val_vrais_inf_ana;
          ptr_col^.res_max := fich_rec_sig_ana^.sig_val_vrais_sup_ana;
          ptr_col^.format := fich_rec_sig_ana^.sig_typ_fmt_res_ana;
          ptr_col^.tar_col := fich_rec_sig_ana^.sig_ind_ana_tarif_ana;
          case ptr_col^.format[1] of
                'D','I': ptr_col^.res_len :=ord (ptr_col^.format[2]) - 46;
                'F':ptr_col^.res_len := ord (ptr_col^.format[2]) - 44;
                'B':ptr_col^.res_len := 5;
                'R':ptr_col^.res_len := ord (ptr_col^.format[2]) - 48 + 1 +
                                        ord (ptr_col^.format[4]) - 48 + 2;
          end;  (* case *)
          if ptr_col^.res_len < 4 then ptr_col^.res_len := 4;

   end;
   close (fich_rec_sig_ana_jbl,error := continue);
   close (fich_rec_sig_ana,error := continue);
```

```
procedure init_tableau(num_jbl:integer);

type prim = record case boolean of
                true :(jbl_ref_pl : packed array[1..10] of char);
                false :(jbl : packed array[1..2] of char;
                        ref : packed array[1..6] of char;
                        pl  : packed array[1..2] of char);
        end;

var clef : prim;
    str : varying[10] of char;
    ptr_lig :^ligne;
    fich_rec_jbl : file of rec_jbl;
    fich_res_temp : file of res_temp;
    temp_jbl : packed array[1..2] of char;
    temp_ref : packed array[1..6] of char;
    ptr_res : ^res;
    i,j,num,x,z,stat : integer;
    num_pl,tentative : integer;
    stop,okopen : boolean;

begin
  ptr_lig := first_ligne;
  open (fich_rec_jbl,'vl_dat:tempjbl.dat',unknown,
        organization := indexed, access_method := keyed, sharing := readwrite,
        error := continue);
  if status(fich_rec_jbl) > 0 then error_sys_stop;
  resetk(fich_rec_jbl,0,error := continue);
  clef.jbl_ref_pl := '0000000000';
  clef.jbl[1] := chr(num_jbl div 10 + 48);
  clef.jbl[2] := chr(num_jbl mod 10 + 48);
  temp_jbl := clef.jbl;
  temp_ref := '000000';
  repeat findk(fich_rec_jbl,0,clef.jbl_ref_pl,geq,error := continue);
                until status(fich_rec_jbl)<=0;
  stop := false;
  i:=0;
  while (not ufb(fich_rec_jbl)) and (not stop) do
  begin
        clef.jbl_ref_pl := fich_rec_jbl^.no_jbl_no_ref_pl_jbl;
        if temp_jbl <> clef.jbl then stop := true
                                else begin
                if temp_ref <> clef.ref then
                begin
                        i:=i+1;
                        creer_ptr_lig(ptr_lig);
                        ptr_lig^.ident := clef.ref;
                        temp_ref := clef.ref;
                        stat := $put_screen(clef.ref,22,1,1);
                        ptr_res := ptr_lig^.res_lig;
                        for num:=1 to nbre_colonne_tableau do
                                        creer_ptr_res(ptr_res);
                end;
                writev(str,clef.pl);
                readv(str,num_pl);
                ptr_lig^.res_dout[num_pl] := true;
                ptr_res := ptr_lig^.res_lig;
                acces_resultat (num_pl,ptr_res);
                ptr_res^.res_type := manquant;
                                end;
                tentative := 0;
                repeat
                  get(fich_rec_jbl,error := continue);
                  okopen := (status(fich_rec_jbl) <= 0);
                  tentative := tentative + 1;
                  if tentative >= nbre_tentative_error
```

```
                    then begin error_sys_more;
                               tentative := 0;
                         end;
            until okopen;
      end; (* while *)
close(fich_rec_jbl,error := continue);
open (fich_res_temp,'vl_online:res_temp',unknown,organization := indexed,
       access_method := keyed, sharing := readwrite,error := continue);
resetk(fich_res_temp,0,error := continue);
clef.jbl_ref_pl := '0000000000';
clef.jbl[1] := chr(num_jbl div 10 + 48);
clef.jbl[2] := chr(num_jbl mod 10 + 48);
tentative := 0;
repeat
 findk(fich_res_temp,0,clef.jbl_ref_pl,geq,error := continue);
 okopen := (status(fich_res_temp) <= 0);
 tentative := tentative + 1;
 if tentative >= nbre_tentative_error then begin error_sys_more;
                                                 tentative := 0;
                                           end;
until okopen;
temp_jbl := clef.jbl;
temp_ref := '000000';
j:=0;
nbre_ligne_tableau := i;
tableau_vide := (i=0);
stop := false;
ptr_lig := first_ligne;
ptr_lig^.num_lig := '000000';
while (not ufb(fich_res_temp)) and (not stop) do
begin
      clef.jbl_ref_pl := fich_res_temp^.rt_jbl_lig_pos;
      if temp_jbl <> clef.jbl then stop := true
                              else begin
            if temp_ref <> clef.ref
              then begin j:=j+1;
              if j>i then begin creer_ptr_lig (ptr_lig);
              ptr_lig^.ident := '000000';
              nbre_ligne_tableau := nbre_ligne_tableau + 1;
              ptr_res := ptr_lig^.res_lig;
              for num := 1 to nbre_colonne_tableau do
                 begin creer_ptr_res(ptr_res);
                       ptr_lig^.res_dout[num] := false;
                 end;
            end
              else  ptr_lig := ptr_lig^.next_lig;
                    ptr_lig^.num_godet := fich_res_temp^.rt_num_pos;
                    ptr_lig^.num_lig := clef.ref;
                    temp_ref := clef.ref;
                                           end;
            writev(str,clef.pl);
            readv(str,num_pl);
            ptr_res := ptr_lig^.res_lig;
            acces_resultat(num_pl,ptr_res);
            modifier_res(ptr_res,ptr_lig,fich_res_temp^.rt_tar,
              fich_res_temp^.rt_pro,fich_res_temp^.rt_res);
            if fich_res_temp^.rt_acc then ptr_res^.res_type := ok;
            ptr_res^.modif := false;
            ptr_res^.res_tar := fich_res_temp^.rt_tar;
            ptr_res^.res_pro := fich_res_temp^.rt_pro;
                              end;
      tentative := 0;
      repeat
       get (fich_res_temp,error := continue);
       okopen := (status(fich_res_temp) <= 0);
       tentative := tentative + 1;
```

```
        if tentative >= nbre_tentative_error then begin error_sys_more;
                                                        tentative := 0;
                                                end;
      until okopen;
   end;
   if (j<i) then begin
                  writev(str,ptr_lig^.num_lig);
                  readv(str,num);
                  for z:=j to i do
                       begin
                          num := num + 1;
                          writev(str,num:1);
                          str := pad('','0',6-length(str)) + str ;
                          ptr_lig := ptr_lig^.next_lig;
                          ptr_lig^.num_godet := 0;
                          readv(str,ptr_lig^.num_lig);
                       end;
                end;

end;    (* of init_tab *)
```

```
procedure heure_date;   (* certainement possible de beaucoup mieux mais !!! *)

var str:packed array[1..11] of char;

begin
   time(str);
   heure_sys[1] := str[1];
   heure_sys[2] := str[2];
   heure_sys[3] := str[4];
   heure_sys[4] := str[5];
   if heure_sys[1] = ' ' then heure_sys[1] := '0';
   if heure_sys[3] = ' ' then heure_sys[3] := '0';

   date (str);
   date_sys[5] := str[1];
   date_sys[6] := str[2];
   date_sys[1] := str[10];
   date_sys[2] := str[11];
   str := substr('0102030405060708091011l2',
                        (index('JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC',
                                substr(str,4,3)) div 3)*2+1,2);
   date_sys[3] := str [1];
   date_sys[4] := str [2];
   if date_sys[5] =' ' then date_sys[5]:='0';
end;    (* of heure_date *)
```

```
procedure init;

var ch:char;
    chspec : spec_char;
    stat,i,j : integer;
    ptr_col : ^colonne;
    ptr_lig : ^ligne;
    ptr_res : ^res;

begin
(* ****************************************** *)
(* initialisation des codes speciaux de res *)

  init_table;

(* ****************************************** *)
(* initialisation des commentaires et erreur*)

  lecture_fich_com;
  lecture_fich_err;

(* ****************************************** *)
(* fenetre d'attente *)

stat := $erase_page(1,1);
entete;
ptr_commande := com_tab[8];
afficher_next_com(ptr_commande);

(* ****************************************** *)
(* initialisation de UC_tab *)

  chspec:=UC_REC;
  for i:=1 to 51 do begin UC_tab[i]:=chspec;
                          chspec:=succ(chspec);
                    end;
  UC_tab[35]:=chspec;

(* ****************************************** *)
(* init du channel pour qiow *)

  stat := $assign ('sys$input',dev_chan);

(* ****************************************** *)
(* init de l'heure et date systeme *)

  heure_date;

(* ****************************************** *)
(* creation des premiers elements du tableau *)

  new(first_ligne);
  first_ligne^.next_lig:=first_ligne;
  first_ligne^.prev_lig:=first_ligne;
  first_ligne^.num_lig := '000000';
  new(first_ligne^.res_lig);
  first_ligne^.res_lig^.next_res:=first_ligne^.res_lig;
  first_ligne^.res_lig^.prev_res:=first_ligne^.res_lig;
  ptr_resultat := first_ligne^.res_lig;
  ptr_resultat^.res_type := premier;
  for i:= 1 to nbre_colonne_tableau do creer_ptr_res(ptr_resultat);

  new(first_colonne);
  first_colonne^.next_col:=first_colonne;
  first_colonne^.prev_col:=first_colonne;
```

```
(* ********************************** *)
(* creation du buffer *)

  new(buffer);
  new(buffer^.res_lig);

        ptr_res:=buffer^.res_lig;
        ptr_res^.res_type := premier;
        ptr_res^.next_res := ptr_res;
        ptr_res^.prev_res := ptr_res;
        for j:=1 to nbre_colonne_tableau do creer_ptr_res(ptr_res);

(* ********************************** *)
(* parametre necessaire devra etre supprimer et remplacer par un parametre
   transferer lors de l'appel *)

  lecture_num_JBL(num_jbl);

(* ********************************** *)
(* initialisation des colonnes du tableau *)

  lecture_colonne ( num_jbl );

(* ********************************** *)
(* initialisation du tableau *)

  init_tableau( num_jbl );

(* ********************************** *)
(* mise a jour des variables globales *)

  ptr_lig := first_ligne;
  high := false;
  if tableau_vide          then begin
                                    fin_pgm := true;
                                    display_error(err_tab[7],1);
                                    lec_touche(ch,chspec);
                                end
                           else fin_pgm := false;
  prem_lig := 1;
  prem_col := 1;
  der_lig := nbre_ligne_tableau;
  der_col := nbre_colonne_tableau;
  prem_lig_fe := 0;
  der_lig_fe := 0;
  prem_col_fe := 0;
  der_col_fe := 0;
  nbre_lig_fe := 10;
  nbre_col_fe := 80 - depcol;
  curs.lig := 1;
  curs.col := 1;
  curs.poscol := depcol;
  ptr_ligne := first_ligne;
  ptr_colonne := first_colonne;
  ptr_resultat := first_ligne^.next_lig^.res_lig;
  ptr_commande := com_tab[1];

(* ********************************** *)
(* presentation des resultats *)

  display_titre ('ACCEPTATION ' + titre_jbl);
  calcul_fe_pt (curs.lig,curs.col);
  ptr_commande := com_tab[1];
  afficher_next_com (ptr_commande);
  set_pos_curs;
```

```
    if tentative >= nbre_tentative_error then begin error_sys_more;
                                                     tentative := 0;
                                                end;
  until okopen;
  fin := false;
  while (not eof(fich_hist)) and (not fin) and
                (fich_hist^.ident_hist_jbl[1] = chdiv)
        and (fich_hist^.ident_hist_jbl[2] = chmod) do
  begin
        existe := recherche_ident (fich_hist^.last_no_ref_hist,ptr_lig);
        if existe then writeln(outf,'sy',' ',fich_hist^.last_no_ref_hist
                                ,' ',ptr_lig^.num_lig,' ');
        tentative := 0;
        repeat
             get(fich_hist,error := continue);
             okopen := (status(fich_hist) <= 0);
             tentative := tentative + 1;
             if tentative >= nbre_tentative_error then begin error_sys_more;
                                                             tentative := 0;
                                                        end;
          until okopen;
  end;
  writeln(outf,str_temp);
  while not eof(inf) do
  begin
        readln(inf,str,error := continue);
        writeln(outf,str,error := continue);
  end;
  close(inf,delete,error := continue);
  close(outf,error := continue);
  close(fich_hist,error := continue);
end;    (* of MAJ_fich_res *)
```

```
tentative := 0;
repeat
   open (fich_hist,'vl_dat:histjbl.dat',unknown,sharing := readwrite,
         organization := indexed, access_method := keyed,error := continue);
   okopen := (status(fich_hist) = 0);
   tentative := tentative + 1;
   if tentative >= nbre_tentative_error then begin error_sys_more;
                                                  tentative := 0;
                                             end;
   if not okopen then close(fich_hist,error:=continue);
until okopen;
resetk(fich_hist,1,error := continue);
tentative := 0;
repeat
   open (inf,'vl_online:param_online',old,error := continue);
   okopen := (status(inf) = 0);
   tentative := tentative + 1;
   if tentative >= nbre_tentative_error then begin error_sys_more;
                                                  tentative := 0;
                                             end;
   if not okopen then close(inf,error:=continue);
until okopen;
reset (inf,error := continue);
open (outf,'vl_online:param_online',new,error := continue);
rewrite (outf,error := continue);
trouve := false; fin := false;
while (not eof(inf)) and (not fin) do
begin
      readln(inf,str,error := continue);
      readv(str,ess,x,y,strres);
      fin := (ess<>'sy') and (trouve);
      if (y=num_jbl) and (strres[1] = '*') then begin ana := ess;
                        trouve:=true;end;
      if (not trouve) then writeln(outf,str,error := continue);
end;
str_temp := str;
fin := false;
num_lig := der_lig;
ptr_lig := first_ligne;
while (num_lig > prem_lig) and (not fin) do
begin
      ptr_lig := ptr_lig^.prev_lig;
      num_col := prem_col;
      ptr_res := ptr_lig^.res_lig;
      while (num_col < der_col) and (not fin) do
      begin
              ptr_res := ptr_res^.next_res;
              fin := fin or ptr_res^.exist;
              num_col := num_col + 1;
      end;
      num_lig := num_lig - 1;
end;
writev(str,ptr_lig^.num_lig);
readv(str,num);
num := num + nbre_ligne_en_suspend;
if trouve then writeln(outf,ana,' ',num:2,' ',num_jbl:2,
                               '**.** num JBL',error := continue);
chdiv := chr(num_JBL div 10 + 48);
chmod := chr(num_JBL mod 10 + 48);
tab := '000000000000';
tab[1] := chdiv;
tab[2] := chmod;
tentative := 0;
repeat
   findk(fich_hist,1,tab,GEQ,error := continue);
   okopen := (status(fich_hist) = 0);
```

```
(*      *       *       * procedure terminaison *       *       *       *)

procedure term(num_jbl:integer);

var fich_rec_ana_res : file of rec_ana_res;
    fich_res : file of res_temp;
    num_lig : integer;
    num_col,tentative : integer;
    continue,okopen : boolean;
    z : integer;


procedure MAJ_fich_res;

type tab16 = packed array[1..6] of char;
     tab12 = packed array[1..2] of char;

var fich_hist : file of rec_hist_jbl;
    inf,outf : text;
    tab : packed array [1..12] of char;
    fin,trouve,existe,okopen : boolean;
    id,num,x,y,tentative : integer;
    num_lig,num_col,stat : integer;
    chmod,chdiv : char;
    ess,ana : packed array [1..2] of char;
    str_temp,str,strres : varying [80] of char;
    ptr_lig : ^ligne;
    ptr_res : ^res;

function recherche_ident(ident : tab16; var ptr_lig : pointeur_ligne):boolean;

var num_lig : integer;
    continue : boolean;

begin
  ptr_lig := first_ligne; num_lig := prem_lig; continue := true;
  while (num_lig <= der_lig + 1) and (continue) do
  begin
        ptr_lig := ptr_lig^.next_lig;
        continue := (ptr_lig^.ident <= ident);
        num_lig := num_lig + 1;
  end;
  ptr_lig := ptr_lig^.prev_lig;
  recherche_ident := (ptr_lig^.ident <= ident);
end;    (* of recherche_ident *)

procedure moins_000;

var num_lig : integer;
    continue : boolean;

begin
  ptr_lig := first_ligne; num_lig := der_lig; continue := true;
  while (num_lig >= prem_lig) and (continue) do
  begin
        ptr_lig := ptr_lig^.prev_lig;
        continue := (ptr_lig^.ident='000000');
        num_lig := num_lig - 1;
  end;
  der_lig := num_lig + 1;
  ptr_lig^.next_lig := first_ligne;
  first_ligne^.prev_lig := ptr_lig;
end;    (* of moins_000 *)

begin
  moins_000;
```

```
var x,i,stat : integer;
    ptr_res : pointeur_res;
    ptr_col : pointeur_colonne;
    str : varying [11] of char;
    tab : packed array [1..11] of char;
    okopen,bool,res_term : boolean;
    tentative : integer;
    strbid,strerr : varying[80] of char;
    realstr : real;

begin
  ptr_res := ptr_lig^.res_lig;
  for i:= 1 to nbre_colonne_tableau do
  begin
        ptr_res := ptr_res^.next_res;
        if (ptr_res^.exist) and (ptr_res^.res_type = ok) and
                (ptr_lig^.ident <> '000000') then
        begin
                acces_colonne (i,ptr_col);
                writev(str,ptr_lig^.ident:6,ptr_col^.nom:5);
                for x:=1 to 11 do tab[x] := str [x];
                for x:=1 to 5 do if tab[x] = ' ' then tab[x] := '0';
                tentative := 0;
                repeat
                  findk(fich_rec_ana_res,1,tab,error:=continue);
                  okopen := (status(fich_rec_ana_res) = 0);
                  tentative := tentative + 1;
                  if tentative >= nbre_tentative_error then
                        begin error_sys_more;
                              tentative := 0;
                        end;
                until okopen;
                fich_rec_ana_res^.ind_ana_tarifiable_res :=
                         ptr_res^.res_tar and ptr_col^.tar_col;
                fich_rec_ana_res^.ind_ana_protocolable_res := ptr_res^.res_pro
                if ptr_res^.res_val < -1 then
                        begin
                        convertir_ecrire(ptr_col^.format,ptr_res^.res_val,
                                        ((not ptr_col^.tar_col) or
                                        (ptr_res^.res_tar)),ptr_res^.res_pro,
                                                strbid);
                        valider_format  (ptr_col^.format,strbid,realstr,bool,
                                                bool,res_term,bool,bool,strerr
                        fich_rec_ana_res^.res_term := res_term;
                        end
                        else fich_rec_ana_res^.res_term := ptr_res^.exist;
                fich_rec_ana_res^.date_intro_res := date_sys;
                fich_rec_ana_res^.heure_intro_res := heure_sys;
                fich_rec_ana_res^.date_real_ana := date_sys;
                fich_rec_ana_res^.heure_real_ana := heure_sys;
                fich_rec_ana_res^.res_num_ana := ptr_res^.res_val;
                if not ufb(fich_rec_ana_res) then
                        update (fich_rec_ana_res,error:=continue);
                (* si ufb alors fichier journal mis a jour.
                        cfr ROULIN      *)
        end;
  end;
end;    (* sauver_ligne_bd *)
```

```
procedure sauver_ligne_fich_res (ptr_lig : pointeur_ligne);
var ptr_res : pointeur_res;
    str : varying [15] of char;
    tab : packed array [1..10] of char;
    i,j : integer;

begin
  ptr_res := ptr_lig^.res_lig;
  for i:= 1 to nbre_colonne_tableau do
  begin
        ptr_res := ptr_res^.next_res;
        if ptr_res^.modif then
        begin
                writev(str,num_jbl:2,ptr_lig^.num_lig,i:2);
                for j:=1 to 10 do if str[j] = ' ' then tab[j] := '0'
                                                  else tab[j] := str[j];
                findk(fich_res,0,tab,error:=continue);
                fich_res^.rt_res := ptr_res^.res_val;
                fich_res^.rt_num_jbl := num_jbl;
                fich_res^.rt_num_col := i;
                writev(str,ptr_lig^.ident);
                readv(str,fich_res^.rt_num_lig);
                fich_res^.rt_num_pos := ptr_lig^.num_godet;
                fich_res^.rt_jbl_lig_pos := tab;
                fich_res^.rt_tar := ptr_res^.res_tar;
                fich_res^.rt_pro := ptr_res^.res_pro;
                fich_res^.rt_acc := ( ptr_res^.res_type = ok );
(* ??? locking *)
if (ufb(fich_res)) and (ptr_res^.exist) then put (fich_res,error:=continue)
 else if (not ufb(fich_res)) and (ptr_res^.exist) then
                update (fich_res,error:=continue)
   else if (not ufb(fich_res)) and not (ptr_res^.exist) then
                delete(fich_res,error:=continue);

        end;
  end;
end;    (* of sauver_ligne_fich_res *)

begin
  ptr_commande := com_tab[11];
  afficher_next_com(ptr_commande);
  tentative := 0;
  repeat
     open (fich_rec_ana_res,'vl_dat:sigres.dat',unknown,
        organization := indexed, access_method := keyed, sharing := readwrite,
        error := continue);
     open (fich_res,'vl_online:res_temp',unknown,organization := indexed,
         access_method := keyed, sharing := readwrite,error := continue);
     okopen := (status(fich_rec_ana_res) <= 0) and (status (fich_res) = 0);
     tentative := tentative + 1;
     if tentative >= nbre_tentative_error then begin error_sys_more;
                                                    tentative := 0;
                                               end;
     if not okopen then begin
                          close(fich_rec_ana_res,error:=continue);
                          close(fich_res,error:=continue);
                        end;
  until okopen;
  resetk(fich_rec_ana_res,1,error := continue);
  resetk(fich_res,0,error := continue);
  num_lig := prem_lig;
  ptr_ligne := first_ligne;
  for z := prem_lig to der_lig do begin
                ptr_ligne := ptr_ligne^.next_lig;
                stat := $put_screen(ptr_ligne^.ident,22,1,1);
                if not partiel then sauver_ligne_bd (ptr_ligne);
```

```
                sauver_ligne_fich_res ( ptr_ligne );
                                end;
    close(fich_rec_ana_res,error := continue);
    close(fich_res,error := continue);

    MAJ_fich_res;

    fin_pgm := true;

end; (* of term *)
```

```
(*      *       *       *       programme       *       *       *       *)

begin
        init;
        while (not fin_pgm) do
        begin
                lec_touche(ch,chspec);
                erase_error;
                set_pos_curs;
                case chspec of
                UC_BS,UC_LF,    (* because of shit rainbow *)
                UC_F12,UC_F13,UC_PF2,UC_PF3,
                UC_INS,UC_SEL,UC_PUP,UC_PDO,
                UC_CUP,UC_CDO,UC_CLE,UC_CRI :begin
                                                deplacement(chspec,high);
                                                high:=false;
                                             end;
                UC_TAU : afficher_next_com(ptr_commande);
                UC_TIL : high := not high;
                UC_PF1 : begin
                                ptr_commande := com_tab[10];
                                chspec := UC_TAU;
                                while chspec=UC_TAU do
                                begin
                                afficher_next_com(ptr_commande);
                                lec_touche(ch,chspec);
                                end;
                                case ch of
                                        'e','E':begin partiel := false;
                                                      term ( num_jbl );
                                                end;
                                        'p','P':begin partiel := true;
                                                      term ( num_jbl );
                                                end;
                                        'f','F':rech_val;
                                        'I','i':intro_auto;
                                        'v','V':correct_auto;
                                        'c','C':(* commentaire *);
                                        'r','R':add_sup_lig_res;
                                        'q','Q':fin_pgm:=true;
                                        'n','N':rech_ident;
                                        '?'    :jump;
                                        'b','B':copy_buffer(ptr_ligne);
                                end;
                                ptr_commande := com_tab[1];
                                afficher_next_com(ptr_commande);
                        end;
                otherwise  mod_add_sup_res(ch,2,ptr_ligne,ptr_resultat,stop);
                end;
                set_pos_curs;
        end;
   pwstat := $erase_page(1,1);
   9999 : pwstat := $erase_page(24,79);
pwstat := $do_command('@menuprinc.com');
end.
```

CONNECTION

==========

Texte des instructions pascal du programme de connection.

```
(*,all data transmited by concentrator have at least this type :
   :: ,E,W,I,CD,CR,LF
No validity control is made on this format.
Attention no use of LF in data.
*)
[inherit ('sys$library:starlet','vl_src:valdem')]

program connection (input,output);
const lmax=100;
      ok=true;
      SOM='<';          (* *)
      EOM=chr(13);      (* *)
      TRA='@';          (* *)
      cl_nb_param = 7;
      hi_nb_param = 40;
      bn_nb_param = 70;

type ligne_de_byte = array[0..lmax] of char;

     reste_char = array[1..lmax-6] of char;

     donnee_corrigee = record
                         pos_zero : char;
                         double_points : array [1..2] of char;
                         error_char : char;
                         warning_char : char;
                         interface_id : array [1..2] of char;
                         data_end : reste_char;
                       end;

     donnee=record case boolean of
                 true:(brut:ligne_de_byte);
                 false:(corrigee:donnee_corrigee);
            end;  (* record *)

vchar5=varying[5] of char;

ligne_transfert = text;

num_lig = [word] -32768..32767;

     form_res = real;

     format_resultat_global = record
        JBL_ligne_col :[key(0)] packed array [1..10] of char;
        resultat : form_res;
        num_JBL:integer;
        num_colonne:integer;
        num_ligne:integer;
        num_pos:integer;
        res_tar : boolean;
        res_pro : boolean;
        res_acc : boolean;
     end;

     es = file of format_resultat_global;

     cl_table_correspondance = array [1..7]  of integer;
     hi_table_correspondance = array [1..40] of integer;
     bn_table_correspondance = array [1..70] of integer;

var buffer:donnee;
    str : varying[5] of char;
    stat,i,ind,id_calc : integer;
    ch : char;
```

```
    okcont,verif : boolean;
    infile,outfile : ligne_transfert;
    ligne_opa0 : num_lig;
    bu : packed array[1..1] of char;
    outfich : es;
    stat_ok, stat_non_ok : integer;

    hi_table:hi_table_correspondance ;
    cl_table:cl_table_correspondance;
    bn_table:bn_table_correspondance;

    hi_ligne_courante : integer;
    cl_ligne_courante : integer;
    bn_ligne_courante : integer;

    JBL_coulter : integer;
    JBL_hitachi : integer;
    JBL_BNA     : integer;

function char_int(inch:char):integer;
begin
   char_int:=ord(inch)-48;
end;

procedure sauver_res_fichier (res: format_resultat_global ; acc : boolean );
(* le fichier est precedement ouvert sinon oblige de l'ouvrir et fermer a
 chaque usage *)
var i:integer;
    okfind : boolean;
    str : varying[10] of char;

begin
   res.res_tar := true; res.res_pro := true;res.res_acc := acc ;
   writev(str,res.num_JBL:2,res.num_ligne:6,res.num_colonne:2);
   for i:=1 to 10 do begin
        if str[i] = ' ' then str[i] := '0';
        res.JBL_ligne_col[i] := str[i];
                    end;
   okfind := false;
   while not okfind do
   begin
     findk(outfich,0,res.JBL_ligne_col,error:=continue);
     okfind := (status(outfich)<=0);
   end;
   outfich^ := res;
   if (res.num_colonne <>0) then
   begin
        if ufb(outfich) then begin
                                put(outfich,error:=continue);
                                findk(outfich,0,res.jbl_ligne_col,
                                        error:=continue);
                             end
                        else update(outfich,error:=continue);
   end;
   unlock(outfich,error:=continue);
end;  (* of sauver_res_fichier *)
```

```
procedure coulter (tab_communication:reste_char);

const  nb_char_res = 5;

type
 vect2=array[1..2] of char;
 vect3=array[1..3] of char;
 vect4=array[1..4] of char;
 vect6=array[1..6] of char;

    result_coulter = record
        cl_result:vect4;
    end;

    reponse_coulter = record
        cl_star:vect4;
        cl_ident:vect6;
        cl_blanc:vect2;
        cl_date:vect3;
        cl_seq:vect3;
        cl_res:array [1..7] of result_coulter;
        cl_check_sum:vect2;      (* unused *)
     end;

     trans = record case boolean of
        true : (data:array[1..lmax] of char);
        false : (coulter:reponse_coulter);
     end;



var transfert : trans;
    res:format_resultat_global;
    ligne_courante :integer ;  (* numero de la prem ligne  *)
    i,x:integer;
    ch:char;

procedure cl_res_in_res(in_res:vect4;var out_res:form_res);
var i:integer;
    str : varying[4] of char;
begin
    str:='';
    for i:=1 to 4 do str := str + in_res[i];
    readv(str,out_res,error:=continue);
end;   (* of cl_res_in_res *)

function tabint(in_tab : vect6):integer;
var res_interm,i:integer;
begin
  res_interm:=0;
  for i:=1 to 6 do  res_interm:=res_interm*10+char_int(in_tab[i]);
  tabint:=res_interm;
end;  (* of tabint *)

                          (* programme *)
begin
for i:=1 to lmax-6 do transfert.data[i] := tab_communication[i];

  res.num_JBL:=JBL_coulter;
  x:= tabint(transfert.coulter.cl_ident);
  res.num_pos:=x;
  if x <> 0 then
  begin
          res.num_ligne := cl_ligne_courante + x;
          for x:= 1 to 7 do
```

```
            begin
              cl_res_in_res(transfert.coulter.cl_res[x].cl_result,res.resultat);
              res.num_colonne:=cl_table[x];
              if x = 1 then res.resultat := res.resultat * 1000;
              sauver_res_fichier(res,false);
            end;
  end;
end;     (* of coulter *)
```

```
procedure hitachi(tab_communication:reste_char);

const first_num = 1;
      last_num = 40;
      hi_max_analyse = 18;
      nb_char_res = 5;                    (* for try *)
      CR=chr(13);
      LF=chr(10);
      nbr_echant_plateau = 40;

type hi_ident=(error,control,normal);
     vect2 = array [1..2] of char;
     vect5 = array [1..5] of char;
     vect3 = array [1..3] of char;

     reponse_hitachi = record
                         hi_error:char;
                         hi_warning:char;
                         hi_code_analyse:vect2;
                         hi_space:array[1..8] of char;
                         hi_identifi:char;
                         hi_num_sequence:vect3;
                         hi_num_plateau:char;
                         hi_num_pos:vect2;
                         hi_blank3:vect2;
                         hi_res:vect5;
                         hi_cr:char;
                         hi_lf:char;
                         hi_etx:char;
                end;

          trans  = record case boolean of
                         true:(data:array[1..lmax-6] of char);
                         false:(hitachi:reponse_hitachi);
                end;


var transfert : trans;
    ligne_courante : integer;
    i,j:integer;
    res:format_resultat_global;
    code_erreur : vect5 ;
    ch:char;

(* function fin_result(res:hitachi_result):boolean;
begin
  fin_result := (res.hi_code_analyse[1]=CR) and (res.hi_code_analyse[2]=LF);
end; (* of fin_result *)

function numcol(num:vect2):integer;
(* utilisation d'une table de correspondance 'numero_analyse/colonne analyse'*
begin
     numcol:=hi_table[char_int(num[1])*10+char_int(num[2])];
     (* uniquement si les numeros d'analyse se suivent sinon utilisation d'une
        table a deux entrees et recherche dichotomique *)
end; (* of numcol *)

procedure hi_res_in_res (in_res:vect5;var out_res: form_res);
(* necessite de connaitre les formats des resultats utilise par 'labo' *)
var i:integer;
    str : varying[5] of char;
begin
    str := '';
    for i:=1 to 5 do str := str + in_res[i];
    ready(str,out_res,error:=continue);
```

```
end; (* of hi_res_in_res *)

                                        (* programme *)

begin (* supposition d'entree : le programme de connection a correctement
         transmis les donnees vers transfert.data  *)

for i:=1 to 5 do code_erreur[i]:='?';                          (* for try *)
for i:=1 to lmax-6 do transfert.data[i] := tab_communication[i];

case transfert.hitachi.hi_identifi of

     'E','C': begin end;(* non implemente *)
        'N' : begin
                res.num_JBL := JBL_hitachi;  (* constante *)
                res.num_pos :=
                char_int(transfert.hitachi.hi_num_plateau)*
                nbr_echant_plateau+
                char_int(transfert.hitachi.hi_num_pos[1])*10+
                char_int(transfert.hitachi.hi_num_pos[2]);
                res.num_ligne := hi_ligne_courante + res.num_pos - 1;
                j:=1;

                hi_res_in_res (transfert.hitachi.hi_res,res.resultat);
                        res.num_colonne := numcol (
                                transfert.hitachi.hi_code_analyse);
                        sauver_res_fichier ( res,false );
                        j:=j+1;
              end;
        end;
end;    (* of hitachi *)
```

```
procedure BNA(tab_communication:reste_char);

const first_num = 1;
      last_num = 40;
      hi_max_analyse = 18;
      nb_char_res = 5;                    (* for try *)
      CR=chr(13);
      LF=chr(10);
      nbr_echant_plateau = 40;

type vect2 = array [1..2] of char;
     vect4 = array [1..4] of char;
     vect5 = array [1..5] of char;
     vect6 = array [1..6] of char;
     vect8 = array [1..8] of char;

     reponse_BNA = record
                        bn_type : char;          (* = r for uni_dir *)
                        bn_ident: vect8;
                        bn_bat  : vect2;
                        bn_norm : char;
                        bn_seq  : vect4;         (* unused *)
                        bn_test : vect2;
                        bn_res  : vect6;
                        bn_flag : char;          (* <,>,E,<blank> *)
                   end;

           trans  = record case boolean of
                        true:(data:array[1..lmax-6] of char);
                        false:(BNA:reponse_BNA);
                   end;


var transfert : trans;
    ligne_courante : integer;
    i,j:integer;
    res:format_resultat_global;
    code_erreur : vect5 ;
    ch:char;
    str : varying[80] of char;
    bool : boolean;
    tabl : array[1..2] of char;

function numcol(num:vect2):integer;
(* utilisation d'une table de correspondance 'numero_analyse/colonne analyse'*)
begin
     numcol:=bn_table[char_int(num[1])*10+char_int(num[2])];
     (* uniquement si les numeros d'analyse se suivent sinon utilisation d'une
        table a deux entrees et recherche dichotomique *)
end; (* of numcol *)

procedure bn_res_in_res (in_res:vect6;var out_res: form_res);
(* necessite de connaitre les formats des resultats utilise par 'labo' *)
var i:integer;
    str : varying[6] of char;
begin
    str := '';
    for i:=1 to 6 do str := str + in_res[i];
    readv(str,out_res,error:=continue);
end; (* of bn_res_in_res *)

function bn_ident_int ( ident : vect8 ) : integer;

var str : varying[8] of char;
    num : integer;
```

```
begin
  str := '';
  for num := 1 to 8 do str := str + ident [num];
  num := 0;
  readv(str,num,error:=continue);
  bn_ident_int := num;
end;    (* of bn_ident_int *)

                                    (* programme *)

begin (* supposition d'entree : le programme de connection a correctement
         transmis les donnees vers transfert.data  *)

for i:=1 to lmax-6 do transfert.data[i] := tab_communication[i];

res.num_pos := bn_ident_int (transfert.BNA.bn_ident);
if res.num_pos <> 0 then
begin
  res.num_JBL := JBL_BNA;  (* constante *)
  res.num_ligne := BN_ligne_courante + res.num_pos - 1;
  bn_res_in_res (transfert.BNA.bn_res,res.resultat);
  res.num_colonne := numcol (transfert.BNA.bn_test);
  bool := true;
  if (transfert.BNA.BN_test[1] = '0') and (transfert.BNA.bn_test[2] = '6') ther
        begin
                sauver_res_fichier ( res,true );
                res.resultat := res.resultat * 0.0295;
                tabl[1] := '3'; tabl[2] := '5';
                res.num_colonne := numcol(tabl);
                sauver_res_fichier ( res,true );
                res.resultat := res.resultat * 120;
                tabl[1] := '3'; tabl[2] := '6';
                res.num_colonne := numcol(tabl);
                sauver_res_fichier ( res,true );
        end
  else if (transfert.BNA.bn_test[1] = '2') and (transfert.BNA.bn_test[2] = '1'
         and  (res.resultat < 5) then begin valider_format('R5.2      ','<5',
                res.resultat,bool,bool,bool,bool,bool,str);
                        sauver_res_fichier(res,true);end
                else if (transfert.BNA.bn_test[1] = '0') and
                        ( (transfert.BNA.bn_test[2] = '1') or
                                (transfert.BNA.bn_test[2] = '2') or
                                        (transfert.BNA.bn_test[2] = '3') )
                           then begin sauver_res_fichier ( res,false );
                                        valider_format('C         ','pr00',
                                                res.resultat,bool,bool
                                                  bool,bool,bool,str);
                                        tabl[1] := '3'; tabl[2] := '7';
                                        res.num_colonne := numcol(tabl);
                                        sauver_res_fichier ( res,true );
                                end
                            else sauver_res_fichier ( res,false );
end;

end;    (* of BNA *)
```

```
(*      *       * procedures de connection UNINA *      *       *)
procedure transparancy(var buffer:donnee);

var i,j:integer;
    tabtrans:array[1..3] of char ;
begin
   tabtrans[1]:=SOM;tabtrans[2]:=EOM;tabtrans[3]:=TRA;
   i:=1;j:=1;
   while i<=lmax do
   begin
      if buffer.brut[i]<>TRA
        then buffer.brut[j]:=buffer.brut[i]
        else begin
                i:=i+1;
                buffer.brut[j]:=tabtrans[char_int(buffer.brut[i])];
             end;
      i:=i+1;j:=j+1;
   end;
   ind:=j-1;
end;  (* of transparancy  *)

function checksum(ligne:ligne_de_byte;ind:integer):boolean;
var str:varying[20] of char;
    compteur:integer;
begin
compteur:=0;
for i:=1 to ind-5 do begin compteur := compteur + ord(ligne[i]);
                     end;
compteur := compteur mod 128;
str:=hex(compteur,2,2);
if ind>=4 then begin
                        checksum:=(str[1]=ligne[ind-4]) and
                                  (str[2]=ligne[ind-3]);
            end
          else checksum := false;
end; (* of checksum *)

procedure envoyer_message_ligne(var line:ligne_transfert;str:vchar5 );
begin
 writeln (line,str);
end; (* of envoyer_message_ligne *)

procedure lecture_char_from_line (var dev_chan:num_lig;var ch:char);
var buf :packed array[1..1] of char;

begin
 stat:=$qiow(,dev_chan,io$_readvblk,,,,buf,1);
 ch:=buf[1];
 end; (* of lecture_char_from_line *)

procedure concat (var ligne:ligne_de_byte;ch:char);
begin (* use of ind:global *)
  ligne[ind]:=ch;
  ind:=ind+1;
end;  (* of concat *)

function fin_ligne(ligne:ligne_de_byte):boolean;
begin (* use of ind:global *)
  fin_ligne:=( (ligne[ind-1]=chr(10)) or (ind>=lmax) );
end;  (* of fin_ligne *)

procedure new_ligne(ligne:ligne_de_byte);
begin (* use of ind:global *)
  ind:=1;
end;  (* of new_ligne *)
```

```
procedure open_ligne;

var dev_name : varying[20] of char;

begin
dev_name:='online';
stat:=$assign(dev_name,ligne_opa0);
open (outfich,'vl_online:res_temp',unknown,sharing := readwrite,
        organization := indexed,access_method := keyed,error := continue);
resetk(outfich,0,error := continue);
open (infile,'online',old,sharing:=readwrite,error:=continue);
                                                    (* open line for output *)
open (outfile,'online',old,sharing:=readwrite,error := continue);
                                                    (* open line for output *)
rewrite(outfile,error:=continue);
reset(infile,error := continue);
okcont := ((status(infile)=0) and (status(outfile)=0) and (status(outfich)=0));
end;  (* of open_ligne  *)

procedure close_ligne;
begin
  close (infile);
  close (outfile);
end; (* of close_line *)

procedure lec_fich_param; (* a modifier *)

var i : integer;
    inf : text;
    str , strres: varying[80] of char;
    tab2 : packed array[1..2] of char;
    y : integer;
    okopen : boolean;

begin
   okopen := false;
   while not okopen do
   begin
        open(inf,'vl_online:param_ONLINE',unknown,sharing:=readwrite,
                                  error := continue);
        okopen := (status(inf)<=0);
        if not okopen then close (inf,error := continue);
   end;
   reset(inf,error := continue);

   readln(inf,str,error := continue);
   readv(str,tab2,hi_ligne_courante,JBL_hitachi,strres,error := continue);

   repeat
     readln(inf,str,error := continue);
     readv(str,tab2,cl_ligne_courante,JBL_coulter,strres,error := continue);
   until tab2<>'sy';

   repeat
     readln(inf,str,error := continue);
     readv(str,tab2,bn_ligne_courante,JBL_bna,strres,error := continue);
   until tab2<>'sy';

   repeat
     readln(inf,str,error:=continue);
     readv(str,tab2,y,hi_table[1],strres,error := continue);  (* 1 ou 2 ??? *)
   until tab2<>'sy';

   for i:= 2 to hi_nb_param do begin
                                  readln(inf,str,error := continue);
```

```
                                        error := continue);
                                end;

    for i:= 1 to cl_nb_param do begin
                                 readln(inf,str,error := continue);
        (* cl *)                 readv(str,tab2,y,cl_table[i],strres,
                                        error := continue);
                                end;

    for i:= 1 to bn_nb_param do begin
                                 readln(inf,str,error := continue);
        (* bn *)                 readv(str,tab2,y,bn_table[i],strres,
                                        error := continue);
                                end;

    close (inf,error := continue);
end;    (* lec_fich_param *)
```

```
                                        (* program *)
begin
okcont := true;
lec_fich_param;          (* si n'existe pas => deconne plein tube *)
open_ligne;
init_table;
stat_ok := 0;                       (* statistique unina *)
stat_non_ok := 0;                   (* statistique unina *)
while ok do                         (* usualy never stopped *)
begin
    new_ligne(buffer.brut);
    buffer.brut[0] := ' ';
    i:=1;                                                  (* for test *)
    while not fin_ligne(buffer.brut) do
    begin
    lecture_char_from_line(ligne_opa0,ch);
    concat(buffer.brut,ch);
    end;
    if buffer.corrigee.warning_char = '1' then lec_fich_param;
    verif:=checksum(buffer.brut,ind);
    if verif=ok then begin
                      stat_ok := stat_ok + 1;
                      (* transparancy(buffer);
                      id_calc := ord(buffer.corrigee.interface_id[1])*256+
                                 ord(buffer.corrigee.interface_id[2]);
                      case id_calc of
                       17228:coulter(buffer.corrigee.data_end);
                       18481:hitachi(buffer.corrigee.data_end);
                       16945:bna    (buffer.corrigee.data_end);
                        end;
                      envoyer_message_ligne(outfile,'$$$');
                      stat:=$qio(,ligne_opa0,io$_readvblk+io$m_purge,,,,bu,0);
                     end

                else begin
                       stat_non_ok := stat_non_ok + 1;
                       envoyer_message_ligne(outfile,'???');
                      end;
end;
close_ligne;

end.
```

MODIF-PARM

==========

Texte des instructions d´un programme permettant de modifier les paramètres des programmes de validation et de connection. Une visualisation des modifications est proposée immédiatement.

```
[inherit ('sys$library:starlet','vl_src:niv0','vl_src:types')]

program modif_param (input,output);

type
        ligne = record
                num :integer;
                ident : packed array[1..2] of char;
                cl : integer;
                rem : varying[80] of char;
                titre : varying[80] of char;
                prev_lig : ^ligne;
                next_lig : ^ligne;
        end;

        pointeur_ligne = ^ligne;

var first_ligne : ^ligne;
    str : varying[80] of char;
    pos,i,j,lig,ind : integer;
    ptr_ligne : ^ligne;
    okmodif,fin,error : boolean;
    pwstat  : integer;
    dev_chan : [word] -32768..32767;
    ch : char;
    nom_ana : packed array[1..5] of char;
    nom_jbl : varying[75] of char;
    fich_ana : file of rec_sig_ana_jbl;
    fich_en_JBL : file of rec_sig_ent_jbl;
(*
[external(lib$put_screen)] function $put_screen
        (out_text : varying[c] of char;
         line : word_integer;
         col : word_integer;
         flag : word_integer := %immed 0) : integer ; extern;
*)

procedure creer_ligne (var ptr_lig : pointeur_ligne);

var temp : ^ligne;

begin
  new (temp);
  temp^.next_lig := ptr_lig^.next_lig;
  temp^.prev_lig := ptr_lig;
  ptr_lig^.next_lig := temp;
  temp^.next_lig^.prev_lig := temp;
  ptr_lig := temp;
end;    (* of creer_ligne *)

procedure lecture_fichier;

var inf : text;
    ptr_lig : ^ligne;
    str,strres : varying[80] of char;

begin
  open(inf,'vl_online:param_online',unknown,sharing:=none,error := continue);
  if status(inf) > 0 then failing_open;
  reset(inf);
  lig := 1;
  ptr_lig := first_ligne;
  okmodif := true;
  while (not eof(inf)) and (okmodif) do
```

```
begin
        creer_ligne (ptr_lig);
        readln(inf,str);
        readv(str,ptr_lig^.ident,ptr_lig^.num,ptr_lig^.cl,strres);
        i := 1 ; j := 1;okmodif := (ptr_lig^.ident <> 'sy' );
        while i<length(strres) do
        begin
                if strres[i] = '.'
                        then begin ptr_lig^.rem := substr(strres,1,i-1);
                                   j := i + 1;
                             end;
                i := i + 1;
        end;
        ptr_lig^.titre := substr (strres,j,length(strres) - j + 1);
        lig := lig + 1;
  end;
  lig := lig - 1;
  close (inf);
end;    (* lecture_fichier *)

procedure sauvetage_fichier;

var outf : text;
    ptr_lig : ^ligne;
    str,strres : varying[80] of char;
    i : integer;

begin
  open(outf,'vl_online:param_online',unknown,error:=continue);
  if status(outf)>0 then failing_open;
  rewrite(outf,error:=continue);
  ptr_lig := first_ligne;
  for i:= 1 to lig do
  begin
        ptr_lig := ptr_lig^.next_lig;
        writeln(outf,ptr_lig^.ident,' ',
                        ptr_lig^.num:2,' ',
                          ptr_lig^.cl:2,
                            ptr_lig^.rem,' .',
                               ptr_lig^.titre,error:=continue);
  end;
  fin := true;
  close (outf,error:=continue);
end;    (* sauvetage_fichier *)

procedure gotoxy(x,y:integer);

var str : varying[10] of char;
    tab : packed array [1..10] of char;
    stat : integer;

begin
  writev(str,chr(27),'[',x:1,';',y:1,'H');
  for i := 1 to length(str) do tab[i] := str[i];
  stat := $qiow(,dev_chan,io$_writevblk,,,,tab,length(str));
end;    (* of gotoxy *)

procedure lecture_char(var ch : char);

var buf : packed array[1..1] of char;
    stat : integer;

begin
  stat := $qiow(,dev_chan,io$_readvblk + io$m_noecho,,,,buf,1);
  ch := buf[1];
end;    (* of lecture_char *)
```

```
procedure afficher_res;

var str,st : varying[80] of char;
    stat : integer;

begin
  writev(st,ptr_ligne^.num:2);
  st := substr(st,1,2);
  writev(str,ptr_ligne^.ident,' ',st:2,' ',ptr_ligne^.cl:2,' ',
         ptr_ligne^.titre,' ':30-length(ptr_ligne^.titre),' ',
                ptr_ligne^.rem:30);
  stat := $put_screen(str,12,1,bolded);

end;    (* of afficher_res *)

procedure modif_comment;

var stat,x : integer;
    modi : boolean;

begin
  modi := true;
  x := 1;
  stat := saisie_edition(ptr_ligne^.titre,x,12,10,30,' ',1,modi,1);
end;    (* of modif_comment *)

procedure verif_mod;

type tab2 = packed array[1..2] of char;

var str:varying[80] of char;
    tab : packed array[1..4] of char;
    x,stat,num_JBL : integer;
    ch : char;

function identJBL (tab :tab2):integer;

var x : integer;
    stop : boolean;
    ptr_lig : ^ligne;

begin
  stop := false;
  if (status(fich_ana) <>0) and (status(fich_en_jbl)<>0) then
                        begin failing_open;okmodif:=false;end;
  ptr_lig := first_ligne;
  x := 1;
  while (x <= lig) and (not stop) do begin
        ptr_lig := ptr_lig^.next_lig;
        if ptr_lig^.ident = ptr_ligne^.ident then
                        begin
                                identJBL := ptr_lig^.cl;
                                stop := true;
                        end;
                                end;
end;    (* of identJBL *)

begin
  num_jbl := identJBl(ptr_ligne^.ident);
  findk(fich_en_JBL,0,num_JBL);
  if ufb(fich_en_JBL) then nom_JBL := ' JBL indefinie '
                      else nom_jbl := fich_en_JBL^.sig_tit_ent_jbl;
  writev(str,num_jbl:2,ptr_ligne^.cl:2);
  for x:=1 to 4 do if str[x] = ' ' then tab[x] := '0'
                                   else tab[x] := str[x];
```

```
  if ufb(fich_ana) then nom_ana := '?????'
                   else nom_ana := fich_ana^.sig_code_ana_jbl;

end;    (* of verif_mod *)

procedure modif_colonne;

var temp,stat,x : integer;
    modi : boolean;
    str : varying[2] of char;
    strtemp : varying[80] of char;

begin
  modi := true;
  writev(str,ptr_ligne^.cl:2);
  stat := saisie_edition(str,x,12,7,2,' ',1,modi,1);
  readv(str,ptr_ligne^.cl,error:=continue);
  if modi then
  begin
    stat := $put_screen('',18,1,normal);
    verif_mod;
    if ptr_ligne^.rem[1] <> '*' then
    begin
    if ptr_ligne^.cl = 0
    then begin
     writeln('Modification de la correspondance :  numero_analyse / analyse, ')
     writeln('le numero d''analyse',ptr_ligne^.num,' n''est pas transfere');
           ptr_ligne^.rem := ' NON_TRANSMIS ';
         end
    else
         begin
    writeln('Modification de la correspondance :  numero_analyse / analyse, ');
    writeln('numero d''analyse :',ptr_ligne^.num,' transfere dans');
    writeln('la JBL :',nom_jbl,' et l''analyse :',nom_ana,'. Press a key');
          ptr_ligne^.rem := ' ';
        end;
    end
    else begin
    writeln('Modification de la JBL, nouvelle JBL : ',nom_JBL,'. Press a key');
           ptr_ligne^.rem := '** JBL **';
         end;
    lecture_char(ch);
    strtemp := pad('',' ',80);
    for x := 18 to 21 do stat := $put_screen(strtemp,x,1,normal);
  end;
end;    (* of modif_comment *)

procedure init;

var i,j : integer;

begin
  new(first_ligne);
  first_ligne^.next_lig := first_ligne;
  first_ligne^.prev_lig := first_ligne;
  first_ligne^.ident := 'ER';
  first_ligne^.titre := 'ERREUR GRAVE !!! TABLEAU VIDE';
  first_ligne^.rem := 'POUSSEZ SUR Q SVP          ';

  lecture_fichier;

  stat := $erase_page(1,1);
  entete;
  display_titre ( 'Modification des paramètres des machines on_line');
  stat := $put_screen('8:up,2:down,E(xit),Q(uit),C(omment),line N(umber)',
```

```
  stat := $put_screen('id nm co                              titre
                              rem',10,1,normal);
  ptr_ligne := first_ligne^.next_lig;
  stat := $assign('sys$input',dev_chan);

  ind:=1;
  fin := not okmodif;

  if not okmodif then stat := $put_screen
     ('   Vous devez purger les fichiers avant modification des paramètres  ',
                 15,1,blinking+bolded)
                  else  afficher_res;
end;    (* of init *)


begin
  open(fich_ana,'vl_dat:siganajbl.dat',readonly,organization := indexed,
        access_method := keyed, sharing := readonly);
  resetk (fich_ana,0);
  open(fich_en_JBL,'vl_dat:sigjbl.dat',readonly,organization := indexed,
        access_method := keyed, sharing := readonly);
  resetk(fich_en_JBL,0);
  init;
  while not fin do
  begin
        lecture_char(ch);
        case ch of '8' : if ind>1 then
                                  begin ptr_ligne := ptr_ligne^.prev_lig;
                                        ind:=ind-1;
                                  end;
                   '2' : if ind<lig then
                                  begin ptr_ligne := ptr_ligne^.next_lig;
                                        ind:=ind+1;
                                  end;
                   'e','E' : sauvetage_fichier;
                   'q','Q' : fin := true;
                   'c','C' : modif_comment;
                   'n','N' : modif_colonne;
         end; (* case *)
         afficher_res;
   end;
  stat := $put_screen('',23,1,0);
  close(fich_ana);
  close(fich_en_jbl);

end.
```

PWSCROLL

Il avait été précisé plus tôt que le programme de validation pourrait être la base de beaucoup d´autre. Le texte du programme qui suit est un programme d´introduction de résultats ayant comme base les procédures du programme de validation.

```
[inherit ('sys$library:starlet','vl_src:valdem','vl_src:types','vl_src:nivl',
          'vl_src:vardem','vl_src:comdem','vl_src:infdem'),
           environment('PWSCROLLmod'),check(NONE)]

module scroll;

type tab2 = packed array[1..2] of char;
     tab6 = packed array[1..6] of char;

procedure pwscroll (tab_num_jbl : tab2 ;prem_ref,dern_ref:tab6);

label 9999;

const nbre_fonction = 12;
      nbre_erreur = 14;
      csi= chr(27);
      char_manquant = '_';
      nbre_char_manquant = 4;
      nbre_char_nom_ana = 5;
      nbre_tentative_error = 10;
      deplig = 7;
      depcol = 10;
      fill_char = '_';

type

        spec_char =(UC_REC,UC_INS,UC_EFF,UC_SEL,UC_PUP,UC_PDO,
                    UC_CUP,UC_CDO,UC_CRI,UC_CLE,
                    UC_PF1,UC_PF2,UC_PF3,UC_PF4,
                    UC_NOT,UC_TAU,
                    UC_F06,UC_F07,UC_F08,UC_F09,UC_F10,
                    UC_LF,
                    UC_F11,UC_F12,UC_F13,UC_F14,
                    UC_BS,
                    UC_F15,UC_F16,
                    UC_RET,
                    UC_F17,UC_F18,UC_F19,UC_F20,
                    UC_TIL,
                    UC_KMI,UC_KCO,UC_KPE,
                    UC_KRI,
                    UC_K00,UC_K01,UC_K02,UC_K03,UC_K04,
                    UC_K05,UC_K06,UC_K07,UC_K08,UC_K09,
                    UC_ESC);

        string = record case boolean of
                true:(str:varying[2000] of char);
                false :(rien : [word] 1..2000;
                        tab : packed array [1..1998] of char);
        end;

        varying_8 = varying [8] of char;
        vary40 = varying[40] of char;

        possible_res_douteux = (ok,anormal,premier,manquant,forrun,superflu,
                                supano);

        colonne = record
                res_min : real;
                res_max : real;
                res_len : integer;
                col_len : integer;
                nom : packed array [1..5] of char;
                format : packed array[1..10] of char;
                num_col : integer;
                tar_col : boolean;
```

```
            prev_col : ^colonne;
        end;

        pointeur_colonne = ^colonne;

        res_douteux = record    (* pour journal des modif de bd *)
                res_dout_ch : char;   (* + , - *)
                res_dout_ident_ana : [key(0)] packed array[1..11] of char;
                res_dout_JBL : integer;
                res_dout_res : real;
        end;

        res_ligne = (* packed *)record  (* des que possible *)
                res_num_col : integer;
                res_val : real;
                res_com : varying[75] of char;
                com_long : boolean;
                com_too_long : boolean;
                res_tar : boolean;
                res_pro : boolean;
                modif : boolean;
                exist : boolean;
                term : boolean;
                res_type :possible_res_douteux;
                next_res : ^res_ligne;
                prev_res : ^res_ligne;
        end;

        pointeur_res = ^res_ligne;

        ligne = record
                ident : packed array[1..6] of char;
                num_lig : packed array[1..6] of char;
                num_godet : integer;
                next_lig : ^ligne;
                prev_lig : ^ligne;
                res_lig : ^res_ligne;
                res_dout : packed array[1..99] of boolean;
        end;

        pointeur_ligne = ^ligne;

        commande = record
                str : varying[80] of char;
                next_com : ^commande;
                prev_com : ^commande;
        end;

        pointeur_commande = ^commande;

        res_temp = record
                rt_jbl_lig_pos :[key(0)] packed array[1..10] of char;
                rt_res : real;
                rt_num_JBL : integer;
                rt_num_col : integer;
                rt_num_lig : integer;
                rt_num_pos : integer;
                rt_tar : boolean;
                rt_pro : boolean;
        end;

(*      p_com = record
                texte : varying[25] of char;
                next : point_com;
                prev : point_com;
```

```
pointeur_p_com = ^p_com; *)

curseur = record
        lig:integer;
        col:integer;
        poscol : integer;
        poslig : integer;
end;

varying_80 = varying[80] of char;
```

```
var    (* global *)

UC_tab : array [1..50] of spec_char;
com_tab : array [0..nbre_fonction] of ^commande;
err_tab : array[1..nbre_erreur] of varying_80;

curs : curseur;

buffer : ^ligne;

ptr_ligne : ^ligne;
ptr_resultat : ^res_ligne;
ptr_colonne : ^colonne;
ptr_commande : ^commande;

first_colonne : ^colonne;
first_ligne : ^ligne;

nbre_ligne_tableau : integer;
nbre_colonne_tableau : integer;

prem_lig : integer;
prem_col : integer;
der_lig : integer;
der_col : integer;

prem_lig_fe : integer;
prem_col_fe : integer;
der_lig_fe : integer;
der_col_fe : integer;

nbre_lig_fe : integer;
nbre_col_fe : integer;

dev_chan : [word] -32768..32767;

titre_jbl : varying [75] of char;

date_sys : packed array [1..6] of char; (* JJMMAA *)
heure_sys : packed array[1..4] of char; (* HHMM *)

num_jbl : integer;

(* var program use *)
ch:char;
chspec:spec_char;
stop:boolean;
pwstat,x,y:integer;
ptr_lig : ^ligne;
pwreel : real;
pwstop,high,fin_pgm,trouve,error : boolean;
```

```
(*      *  procedure creation et gestion des pointeurs  *       *       *)

procedure creer_ptr_lig (var ptr_lig:pointeur_ligne);

var temp_lig : ^ligne;
    ptr_res : ^res_ligne;
begin
  new(temp_lig);
  temp_lig^.prev_lig := ptr_lig;
  temp_lig^.next_lig := ptr_lig^.next_lig;
  ptr_lig^.next_lig := temp_lig;
  temp_lig^.next_lig^.prev_lig := temp_lig;
  new(temp_lig^.res_lig);
(* init for first res in the line *)
  ptr_res := temp_lig^.res_lig;
  ptr_res^.next_res :=ptr_res;
  ptr_res^.prev_res := ptr_res;
  ptr_res^.res_num_col := 0;
  ptr_res^.res_type := premier;
  ptr_lig:=temp_lig;
end;    (* of creer_ptr_lig *)


procedure creer_ptr_res (var ptr_res : pointeur_res );

var temp_res : ^res_ligne;

begin
  new(temp_res);
  temp_res^.prev_res := ptr_res;
  temp_res^.next_res := ptr_res^.next_res;
  ptr_res^.next_res := temp_res;
  temp_res^.next_res^.prev_res := temp_res;
  ptr_res:=temp_res;                    (* init possible via un cast *)
  ptr_res^.res_type := ok;
  ptr_res^.res_pro := true;
  ptr_res^.res_tar := true;
  ptr_res^.term := false;
  ptr_res^.res_com := '';
  ptr_res^.com_long := false;
  ptr_res^.com_too_long := false;
  ptr_res^.modif := false;
  ptr_res^.exist := false;
end;    (* of creer_ptr_res *)


procedure creer_ptr_col (var ptr_col : pointeur_colonne );

var temp_col : ^colonne;

begin
  new(temp_col);
  temp_col^.prev_col := ptr_col;
  temp_col^.next_col := ptr_col^.next_col;
  ptr_col^.next_col := temp_col;
  temp_col^.next_col^.prev_col := temp_col;
  ptr_col:=temp_col;
end;    (* of creer_ptr_col *)


procedure creer_ptr_com (var ptr_com : pointeur_commande );

var temp_com : ^commande;
```

```
  new(temp_com);
  temp_com^.prev_com := ptr_com;
  temp_com^.next_com := ptr_com^.next_com;
  ptr_com^.next_com := temp_com;
  temp_com^.next_com^.prev_com := temp_com;
  ptr_com:=temp_com;
end;    (* of creer_ptr_com *)

procedure creer_p_com(var ptr:point_com);       (* <> des precedents *)

var temp : point_com;

begin
  new(temp);
  temp^.prec := ptr;
  temp^.next := ptr^.next;
  ptr^.next := temp;
  ptr := temp;
end;    (* of creer_p_com *)

procedure suppr_ptr_res (var ptr_res : pointeur_res);

var fin_res,temp_res : ^res_ligne;

begin
  fin_res := ptr_res;   (* !!! fin_res devient un pointeur pendant *)
  repeat
        temp_res := ptr_res^.next_res;
        dispose(ptr_res);
        ptr_res := temp_res;
  until (ptr_res = fin_res);
end;    (* of suppr_ptr_res *)

procedure suppr_ptr_lig (var ptr_lig : pointeur_ligne);

var fin_lig,temp_lig : ^ligne;

begin
  fin_lig := ptr_lig;   (* !!! fin_lig devient un pointeur pendant *)
  repeat
        temp_lig := ptr_lig^.next_lig;
        suppr_ptr_res (ptr_lig^.res_lig);
        dispose(ptr_lig);
        ptr_lig := temp_lig;
  until (ptr_lig = fin_lig);
end;    (* of suppr_ptr_lig *)

procedure suppr_ptr_col (var ptr_col : pointeur_colonne);

var fin_col,temp_col : ^colonne;

begin
  fin_col := ptr_col;   (* !!! fin_col devient un pointeur pendant *)
  repeat
        temp_col := ptr_col^.next_col;
        dispose(ptr_col);
        ptr_col := temp_col;
  until (ptr_col = fin_col);
end;    (* of suppr_ptr_col *)
```

```
procedure acces_colonne (num:integer;var ptr_col : pointeur_colonne);

var i:integer;

begin
  ptr_col := first_colonne;
  for i:=1 to num do ptr_col := ptr_col^.next_col;
end;    (* of acces_colonne *)


procedure acces_ligne (num:integer;var ptr_lig : pointeur_ligne);

var i:integer;

begin
  ptr_lig:=first_ligne;
  for i:=1 to num do ptr_lig := ptr_lig^.next_lig;
  curs.lig:=num;
end;    (* of acces_ligne *)


procedure acces_resultat (num:integer;var ptr_res : pointeur_res);

var i:integer;

begin
  for i:=1 to num do ptr_res := ptr_res^.next_res;
  curs.col:=num;
end;    (* of acces_resultat *)


procedure find_res (x,y:integer);

begin
  if (x<=nbre_ligne_tableau) and (x>0) and (y<=nbre_colonne_tableau) and (y>0)
        then begin
                acces_ligne(x,ptr_ligne);
                ptr_resultat := ptr_ligne^.res_lig;
                acces_resultat (y,ptr_resultat);
                acces_colonne (y,ptr_colonne);
             end
        else begin (* attention au tab vide *) end;
end;    (* of find_res *)
```

```
(*      *       *  quelques procedures tressss utiles   *       *       *)
procedure lec_touche(var touche:char; var special:spec_char);

label  9;

var x,stat : integer;
    toto : packed array [1..1] of char;

(* UC_   : unprintable char *)
begin
   touche := chr(0);
   special := UC_not;
   stat := $qiow(,dev_chan,io$_readvblk+io$m_noecho, , , ,toto,1);
   case ord(toto[1]) of
        27:begin
             stat:=$qiow(,dev_chan,io$_readvblk+io$m_noecho, , , ,toto,1);
             if not(toto[1] in [chr(27),'[','0']) then goto 9;
             if toto[1] = chr(27) then special:=UC_ESC
             else
             begin
               stat:=$qiow(,dev_chan,io$_readvblk+io$m_noecho, , , ,toto,1);
               case toto[1] of
               'P','Q','R','S' : special := UC_tab[ord(toto[1])-69];
               'A','B','C','D' : special := UC_tab[ord(toto[1])-58];
               'l','m','n','o','p','q','r','s','t',
               'u','v','w','x','y' : special := UC_tab[ord(toto[1])-71];
               'M' : special := UC_RET;
               '0','1','2','3','4','5','6','7','8','9': begin
           x:=ord(toto[1])-48;
           stat :=$qiow(,dev_chan,io$_readvblk+io$m_noecho, , , ,toto,1);
           if not (toto[1] in ['0'..'9','~']) then goto 9;
           if (toto[1] in ['0'..'9']) then begin
                x:=x*10+ord(toto[1])-48;
                stat :=$qiow(,dev_chan,io$_readvblk+io$m_noecho, , , ,toto,1);
                if toto[1] <> '~' then goto 9;
                 end;
           if not (x in [1..6,17..19,20,21,23..26,28,29,31..34]) then goto 9;
                   special := UC_tab[x];
                end;
                 otherwise goto 9;
                end; (* of case toto[1] *)
              end; (* of if *)
            end; (* of case 27 *)
        8  : special := UC_BS;
        9  : begin touche := toto[1];special := UC_TAU; end;
        10 : special := UC_LF;
        13 : begin special := UC_RET; touche := toto[1]; end;
  39,126,96: begin special := UC_TIL; touche := toto[1]; end;
 otherwise   touche:=toto[1];
        end; (* of case *)
9: ;
end;    (* of lec_touche *)


procedure afficher_next_com (var ptr_com : pointeur_commande);

var stat : integer;

begin
        ptr_com := ptr_com^.next_com;
        if substr(ptr_com^.str,1,10) = 'NO COMMENT' then ptr_com
                                    := ptr_com^.next_com;
        stat := $put_screen(ptr_com^.str,23,1,3);
end;    (* of afficher_next_com *)
```

```
procedure error_sys_stop;

begin
  smg$ring_bell(displgen,1);
  failing_open;
  goto 9999;
end;    (* of error_sys_stop *)


procedure error_sys_more;

var ch : char;
    chspec : spec_char;

begin
  smg$ring_bell(displgen,1);
  display_error(err_tab[8],1);
  lec_touche(ch,chspec);
  case ch of 'q','Q' : goto 9999;
  end; (* of case *)
  erase_error;
end;    (* of error_sys_more *)

function str40(var str:varying[d] of char):vary40;

var trouve : boolean;
    x : integer;

begin
  x := index(str,'œ');
  if x > 0 then
  begin
        str40 := substr(str,1,x-1);
        if x < length(str) then str := substr(str,x+1,length(str)-x)
                           else str := '';
  end
  else
  begin
    if length(str) > 40 then
    begin
        trouve := false;
        x:=41;
        repeat
                trouve :=(str[x]='&');
                x := x - 1;
        until (x = 0) or (trouve);
        str40 := substr(str,1,x);
        str := substr(str,x+1,length(str)-x);
    end
    else begin str40 := str; str:=''; end;
  end;
end;    (* of str40 *)

procedure charger_com;
var pt_trav,bid : point_com;
    point_co : integer;
    str : varying[11] of char;
    tab : packed array[1..11] of char;
    x : integer;
    okopen : boolean;
    tentative : integer;

begin
writev(str,ptr_ligne^.ident:6,ptr_colonne^.nom:5);
tab := str ;
```

```
tentative := 0;
repeat
  findk(fich_ana_res,1,tab,error:=continue);
  okopen := (status(fich_ana_res) <= 0);
  tentative := tentative + 1;
  if tentative >= nbre_tentative_error then begin error_sys_more;
                                                  tentative := 0;
                                            end;
until okopen;

init_com := nil;
pt_trav := nil;
point_co :=fich_ana_res^.pt_res_com;
while point_co > 0 do
   begin
   find(fich_ext_res_ana,point_co);
   new(bid);
   bid^.texte := fich_ext_res_ana^.txt_ext_res_ana;
   bid^.next := nil;
   bid^.prec := nil;
   if init_com = nil then
      begin
      init_com := bid;
      pt_trav := bid;
      end
   else
      begin
      pt_trav^.next := bid;
      bid^.prec := pt_trav;
      pt_trav := pt_trav^.next
      end;
   point_co := fich_ext_res_ana^.pt_next_ext_res_ana
   end;
   unlock(fich_ext_res_ana,error:=continue);
end;


procedure decharger_com;
var point,point_co:integer;
    pt_trav : point_com;

    str : varying[11] of char;
    tab : packed array[1..11] of char;
    x : integer;
    okopen : boolean;
    tentative : integer;

begin
      writev(str,ptr_ligne^.ident:6,ptr_colonne^.nom:5);
      tab := str ;
      for x:=1 to 5 do if tab[x] = ' ' then tab[x] := '0';
      tentative := 0;
      repeat
        findk(fich_ana_res,1,tab,error:=continue);
        okopen := (status(fich_ana_res) <= 0);
        tentative := tentative + 1;
        if tentative >= nbre_tentative_error then begin error_sys_more;
                                                        tentative := 0;
                                                  end;
      until okopen;

      if init_com <> nil then fich_ana_res^.res_term := true;
      point_co := fich_ana_res^.pt_res_com;
      while point_co <> 0 do
         begin
```

```
         point := point_co;
         point_co := fich_ext_res_ana^.pt_next_ext_res_ana;
         rel_rec_ext_res_ana(point);
         end;
      fich_ana_res^.pt_res_com := 0;
      pt_trav := init_com;
      point_co := 0;
      while pt_trav <> nil do
         begin
         if length(pt_trav^.texte) > 0 then
            begin
            new_rec_ext_res_ana(point);
            find(fich_ext_res_ana,point);
            if length(pt_trav^.texte) > longcom - 4 then
               begin
               pt_trav^.texte := substr(pt_trav^.texte,1,longcom-4);
               end;
            fich_ext_res_ana^.txt_ext_res_ana := pt_trav^.texte;
            fich_ext_res_ana^.pt_next_ext_res_ana := 0;
            update(fich_ext_res_ana);
            if fich_ana_res^.pt_res_com = 0 then
               fich_ana_res^.pt_res_com := point
            else
               begin
               find(fich_ext_res_ana,point_co);
               fich_ext_res_ana^.pt_next_ext_res_ana := point;
               update(fich_ext_res_ana)
               end;
            point_co := point;
            end;
         pt_trav := pt_trav^.next;
         end;
      fich_ana_res^.res_prot := false;
      repeat update(fich_ana_res,error:=continue);until status(fich_ana_res)<=0
      unlock(fich_ana_res,error:=continue);
end;

function long(* ? *)(ptr_res:pointeur_res):boolean;

var x : integer;
    trouve : boolean;
    tab : packed array[1..7] of char;
    str : varying[80] of char;

begin
  str := ptr_res^.res_com;
  x := length(ptr_res^.res_com);
  if (x<2) or (ptr_res^.com_too_long)
        then long := false
        else begin
                repeat
                        x:= x-1;
                        trouve := (str[x] = '&') or (str[x] = 'œ');
                until (trouve) or (x=1);
                if not (trouve) or (str[x]='œ') then long := false
                                        else begin
                                tab := substr(str,x+1,length(str) - x - 1);
                                repeat findk(fich_rem_std,0,tab,geq,
                                        error := continue)
                                until status(fich_rem_std) <= 0;
                                tab := fich_rem_std^. code_no_rem_std;
                                long := (tab[6]<>'0') or (tab[7]<>'0');
                                                end;
            end;
end;    (* of long *)
```

```
procedure fenetre_commentaire; forward;
```

```
(*      *       *       *       procedure ecran *       *       *       *)

function gitixy(x,y:integer):varying_8;

var str:varying[8] of char;
    i:integer;

begin
writev(str,csi,'[',x:1,';',y:1,'H');
gitixy := str;
end;    (* of gitixy *)


procedure make_res_affich(i,j:integer;ptr_res:pointeur_res;
                                  var str_res:varying[c] of char);
var ptr_col : ^colonne;
   num, z : integer;
   str : varying [20] of char;

begin
i:=deplig + i - prem_lig_fe;
num:=depcol;
acces_colonne(prem_col_fe,ptr_col);
for z:=prem_col_fe to (j-1) do
begin
  num := num + ptr_col^.col_len;
  ptr_col := ptr_col^.next_col;
end;
(* num := num + (ptr_col^.col_len - ptr_col^.res_len - 1) div 2; *)
j:=num;
str := '';
str_res := gitixy(i,j) + pad('',' ',ptr_colonne^.res_len);
if ptr_res^.exist then begin
        convertir_ecrire(ptr_col^.format,ptr_res^.res_val,
                         ((not ptr_col^.tar_col) or ( ptr_res^.res_tar)),
                          ptr_res^.res_pro,str);
        str := pad(str,' ',ptr_col^.res_len);
        writev(str_res,csi,'[0;',ord(ptr_res^.res_type):1,'m',str,csi,'[0m');
        str_res :=  gitixy (i,j) + str_res;
                              end;
if ptr_res^.res_type = manquant then str_res:= gitixy (i,j) +
                                      pad('',char_manquant,ptr_colonne^.res_len);
if ptr_res^.res_com <>'' then str_res := str_res + '¢'
                         else str_res := str_res + ' ';
end;    (* of make_res_affich *)


procedure affich_format(ptr_col : pointeur_colonne);
var stat : integer;
    str : varying[10] of char;
begin
  case ptr_col^.format[1] of
  'D','I','d': str := pad('','*',ord(ptr_col^.format[2])-48);
  'F','f' : str := '1/' + pad('','*',ord(ptr_col^.format[2])-48);
  'B','b' : str := '***';
  'R','r' : str := pad('','*',ord(ptr_col^.format[2])-48) + '.' +
                   pad('','*',ord(ptr_col^.format[4])-48);
  'C','c' : str := '_____';
  'A','P','S','G','a','p','s','g': str := pad('','*',ptr_col^.res_len);
  end; (* of case *)
  stat := $put_screen(str,23,71,3);
end;    (* of affich_format *)

procedure affich_resultat;
```

```
    buf : string;

begin
   make_res_affich(curs.lig,curs.col,ptr_resultat,buf.str);
   stat := $qiow( ,dev_chan,io$_writevblk, , , ,buf.tab,length(buf.str));
end;    (* of affich_resultat *)


procedure affich_fenetre;


var buf_com:string;
    i,j:integer;
    str_res : varying[30] of char;
    ptr_col : ^colonne; ptr_lig : ^ligne; ptr_res : ^res_ligne;
    stat : integer;
    temp_curs : curseur;
    pla_col : integer;

begin
  temp_curs := curs;
  buf_com.str := '';
  pla_col := depcol;
  ptr_col := ptr_colonne;
  ptr_lig := ptr_ligne;
  ptr_res := ptr_resultat;
  acces_colonne (prem_col_fe,ptr_colonne);
  buf_com.str := gitixy(deplig-2,0) + csi + '[K' + '  ref  ';
  for i:= prem_col_fe to der_col_fe do
  begin
       writev(str_res,ptr_colonne^.nom:5);
       buf_com.str := buf_com.str + gitixy (deplig-2,pla_col +
              (ptr_colonne^.col_len - nbre_char_nom_ana ) div 2) + str_res;
       pla_col := pla_col + ptr_colonne^.col_len;
       ptr_colonne := ptr_colonne^.next_col;
  end;
  buf_com.str := buf_com.str + gitixy(deplig-1,0) + csi + '(0' +
                 pad('',chr(115),80) + csi +'(B';
  for i:=prem_lig_fe to der_lig_fe do
  begin
       acces_colonne (prem_col_fe,ptr_colonne);
       find_res(i,prem_col_fe);
       writev(str_res,ptr_ligne^.ident:6,' ',csi,'(0',chr(120),csi,'(B');
       buf_com.str := buf_com.str + gitixy(i-prem_lig_fe+deplig,0)+csi +
                      '[K' + str_res;
       for j:=prem_col_fe to der_col_fe do
       begin
              make_res_affich(i,j,ptr_resultat,str_res);
              buf_com.str := buf_com.str + str_res;
              ptr_resultat := ptr_resultat^.next_res;
              ptr_colonne := ptr_colonne^.next_col;
       end;
       stat := $qiow( ,dev_chan,io$_writevblk, , , ,buf_com.tab,
                      length(buf_com.str));
       buf_com.str := '';
   end;
   ptr_colonne := ptr_col;
   ptr_ligne := ptr_lig;
   ptr_resultat := ptr_res;
   curs := temp_curs;

end;    (* of affich_fenetre *)


procedure calcul_pos_curs;
```

```
var i:integer;
    ptr_col : ^colonne;

begin
  acces_colonne(prem_col_fe,ptr_col);
  curs.poscol := depcol;
  for i:= prem_col_fe to (curs.col - 1) do
  begin
    curs.poscol := curs.poscol + ptr_col^.col_len;
    ptr_col := ptr_col^.next_col;
  end;
end;    (* of calcul_pos_curs *)


procedure sup_pos_curs;

var buf_com : string;
    stat : integer;
    str : varying[20] of char;

begin
   str := gitixy(curs.lig - prem_lig_fe + deplig,curs.poscol - 1);
   buf_com.str := str + ' ' + str;
   stat := $qiow( ,dev_chan,io$_writevblk, , , ,buf_com.tab,
                        length(buf_com.str));
end;    (* of sup_pos_curs *)


procedure set_pos_curs;

var buf_com : string;
    stat : integer;

begin
   buf_com.str := ' ' + gitixy (curs.lig - prem_lig_fe + deplig,curs.poscol - 1
                  + '>' + chr(8);
   stat := $qiow( ,dev_chan,io$_writevblk, , , ,buf_com.tab,
                        length(buf_com.str));
end;    (* of position curseur *)

 procedure calcul_fe_pt(x,y:integer); forward;
                (* normalement curs.lig,curs.col *)
 procedure calcul_fe_pt_intro(x,y:integer); forward;
                (* normalement curs.lig,curs.col *)
```

```
(*      *       *   procedure de deplacement   *       *       *)

procedure deplacement (special : spec_char; high_deplac : boolean);

var i,j,long : integer;
    ptr_temp,ptr_col : ^colonne;
    sup_i : boolean;
    old_lig, old_col : integer;

begin
 old_lig := prem_lig_fe; old_col := der_col_fe;
 if (not high_deplac) then begin
        case special of
        (* for deplacement curseur *)
        UC_CUP:if curs.lig > prem_lig then begin
                        curs.lig:=curs.lig-1;
                        ptr_ligne := ptr_ligne^.prev_lig;
                        ptr_resultat := ptr_ligne^.res_lig;
                        acces_resultat(curs.col,ptr_resultat);
                        if curs.lig <prem_lig_fe then deplacement(UC_INS,false
                                                end
                                        else display_error(err_tab[1],1);
        UC_CDO:if curs.lig < der_lig then begin
                        curs.lig:=curs.lig+1;
                        ptr_ligne := ptr_ligne^.next_lig;
                        ptr_resultat := ptr_ligne^.res_lig;
                        acces_resultat(curs.col,ptr_resultat);
                        if curs.lig >der_lig_fe then deplacement(UC_PUP,false)
                                                end
                                        else display_error(err_tab[2],1);

        UC_CLE:if curs.col > prem_col then begin
                        curs.col := curs.col - 1;
                        ptr_colonne := ptr_colonne^.prev_col;
                        curs.poscol := curs.poscol - ptr_colonne^.col_len;
                        ptr_resultat := ptr_resultat^.prev_res;
                        if curs.col< prem_col_fe then deplacement(UC_SEL,false
                                                end
                                        else display_error(err_tab[3],1);

        UC_CRI:if curs.col < der_col then begin
                        curs.col := curs.col + 1;
                        curs.poscol := curs.poscol + ptr_colonne^.col_len;
                        ptr_resultat := ptr_resultat^.next_res;
                        ptr_colonne := ptr_colonne^.next_col;
                        if curs.col > der_col_fe then deplacement(UC_PDO,false
                                                end
                                        else display_error(err_tab[4],1);

        (* for deplacement fenetre *)
        UC_BS,UC_F12,UC_INS:begin
                        prem_lig_fe := prem_lig_fe - nbre_lig_fe;
                        if prem_lig_fe < prem_lig then prem_lig_fe := prem_lig;
                        der_lig_fe := prem_lig_fe + nbre_lig_fe - 1;
                        if der_lig_fe > der_lig then der_lig_fe := der_lig;
                        if curs.lig > der_lig_fe then curs.lig := der_lig_fe;
                        find_res(curs.lig,curs.col);
                        if (old_lig <> prem_lig_fe) or (old_col <> der_col_fe)
                                        then affich_fenetre
                                        else display_error(err_tab[1],1);
                      end;
        UC_PF2,UC_SEL:begin
                        acces_colonne(prem_col_fe,ptr_col);
                        ptr_temp := ptr_col;
                        i := prem_col_fe - 1;
```

```
        long :=0;
        ptr_col := ptr_col^.prev_col;
        while (i>= prem_col) and (long<=nbre_col_fe) do
        begin
                long := ptr_col^.col_len + long;
                i := i-1;
                ptr_col := ptr_col^.prev_col;
        end;
        ptr_col := ptr_temp;
        prem_col_fe := i + 1 + ord(long>nbre_col_fe);
        sup_i := (long > nbre_col_fe);
        i := j;
        while (i<=nbre_colonne_tableau) and (long<=nbre_col_fe) do
        begin
                long := ptr_col^.col_len + long;
                i := i + 1;
                ptr_col := ptr_col^.next_col;
        end;
        der_col_fe := i - 1 - ord(long>nbre_col_fe) + ord(sup_i);
        if curs.col > der_col_fe then curs.col := der_col_fe;
        calcul_pos_curs;
        find_res(curs.lig,curs.col);
        if (old_lig <> prem_lig_fe) or (old_col <> der_col_fe)
                then affich_fenetre
                else display_error(err_tab[3],1);
      end;
UC_LF,UC_F13,UC_PUP:begin
        der_lig_fe := der_lig_fe + nbre_lig_fe;
        if der_lig_fe > der_lig then der_lig_fe := der_lig;
        prem_lig_fe := der_lig_fe - nbre_lig_fe + 1;
        if prem_lig_fe < prem_lig then prem_lig_fe := prem_lig;
        if curs.lig < prem_lig_fe then curs.lig := prem_lig_fe;
        find_res(curs.lig,curs.col);
        if (old_lig <> prem_lig_fe) or (old_col <> der_col_fe)
                then affich_fenetre
                else display_error(err_tab[2],1);
       end;
UC_PF3,UC_PD0:begin
        acces_colonne(der_col_fe,ptr_col);
        ptr_temp := ptr_col;
        i := der_col_fe + 1;
        j := der_col_fe;
        long :=0;
        ptr_col := ptr_col^.next_col;
        while (i<=nbre_colonne_tableau) and (long<=nbre_col_fe) do
        begin
                long := ptr_col^.col_len + long;
                i := i + 1;
                ptr_col := ptr_col^.next_col;
        end;
        ptr_col := ptr_temp;
        der_col_fe := i - 1 - ord(long>nbre_col_fe);
        sup_i := (long > nbre_col_fe);
        i := j;
        while (i>= prem_col) and (long<=nbre_col_fe) do
        begin
                long := ptr_col^.col_len + long;
                i := i-1;
                ptr_col := ptr_col^.prev_col;
        end;
        prem_col_fe := i + 1 + ord(long>nbre_col_fe) - ord(sup_i);
        if curs.col < prem_col_fe then curs.col := prem_col_fe;
        calcul_pos_curs;
        find_res(curs.lig,curs.col);
        if (old_lig <> prem_lig_fe) or (old_col <> der_col_fe)
```

```
                                else display_error(err_tab[4],1);
            end;
        end; (* of case *)
                end  (* of then *)
              else begin
        case special of
        (* for high deplacement curseur *)
        UC_CUP:curs.lig := prem_lig_fe;
        UC_CDO:curs.lig := der_lig_fe;
        UC_CLE:curs.col := prem_col_fe;
        UC_CRI:curs.col := der_col_fe;

        (* for high deplacement fenetre *)
        UC_F12,UC_BS,UC_INS:curs.lig := prem_lig;
        UC_PF2,UC_SEL:curs.col := prem_col;
        UC_F13,UC_LF,UC_PUP:curs.lig := der_lig;
        UC_PF3,UC_PDO:curs.col := der_col;
        end; (* of case *)
        find_res (curs.lig,curs.col);
        calcul_pos_curs;
        if (curs.lig > der_lig_fe) or (curs.lig < prem_lig_fe) or
             (curs.col > der_col_fe) or (curs.col < prem_col_fe) then
                calcul_fe_pt(curs.lig,curs.col);

                end;  (* of else *)

end;    (* of deplacement *)
```

```
procedure calcul_fe_pt (* (x,y:integer) *);  (* normalement curs.lig,curs.col *

var long : integer;
    ptr_col : ^colonne;

begin
  prem_lig_fe := x - (nbre_lig_fe div 2);
  if prem_lig_fe < prem_lig then prem_lig_fe := prem_lig;
  der_lig_fe := prem_lig_fe + nbre_lig_fe - 1;
  if der_lig_fe > der_lig then der_lig_fe := der_lig;
  prem_lig_fe := der_lig_fe - nbre_lig_fe + 1;
  if prem_lig_fe < prem_lig then prem_lig_fe := prem_lig;

  acces_colonne( y , ptr_col);
  long := - (ptr_col^.col_len div 2);
  while ( y >= prem_col ) and (long < nbre_col_fe div 2) do
  begin
        long := long + ptr_col^.col_len;
        ptr_col := ptr_col^.prev_col;
        y := y - 1;
  end;
  der_col_fe := y + ord(long > (nbre_col_fe) div 2);
  deplacement (UC_PDO,false);
end;    (* of calcul_fe_pt *)

procedure calcul_fe_pt_intro (* (x,y:integer) *);

var long : integer;
    ptr_col : ^colonne;

begin
  prem_lig_fe := x - 1;
  if prem_lig_fe < prem_lig then prem_lig_fe := prem_lig;
  der_lig_fe := prem_lig_fe + nbre_lig_fe - 1;
  if der_lig_fe > der_lig then der_lig_fe := der_lig;
  prem_lig_fe := der_lig_fe - nbre_lig_fe + 1;
  if prem_lig_fe < prem_lig then prem_lig_fe := prem_lig;

  acces_colonne( y , ptr_col);
  long := - (ptr_col^.col_len div 2);
  while ( y >= prem_col ) and (long < nbre_col_fe div 2) do
  begin
        long := long + ptr_col^.col_len;
        ptr_col := ptr_col^.prev_col;
        y := y - 1;
  end;
  der_col_fe := y + ord(long > (nbre_col_fe) div 2);
  deplacement (UC_PDO,false);
end;    (* of calcul_fe_pt_intro *)
```

```
(*      *       *       *       procedure ???   *       *       *       *)

procedure instring (var str:varying[c] of char;num,lig,col,mode:integer;
                        fill_char:char;x:integer;var modif:boolean;y:integer;
                        var fin_intro:boolean);

var sais : integer;
    fin : boolean;
    str_com : varying [75] of char;
    ptr_temp : pointeur_commande;

begin
 fin := (mode<=0);
 if (length(str)>mode) and (mode<>0) then str := substr(str,1,mode);
 while not fin do   (* important si <esc> peut devenir un terminator alors
                                 possibite de la fenetre de commentaire *)
 begin
   modif := true;
   sais := saisie_edition(str,num,lig,col,mode,fill_char,x,modif,y);
   fin := ((sais = 10) or (sais = -1));
   if (sais =  1) or (sais = 5)
                then begin ptr_temp := ptr_commande ;  (* beaucoup mieux si
                                                                    esc *)
                           fenetre_commentaire;
                           ptr_commande := ptr_temp;
                           afficher_next_com(ptr_commande);
                           str_com := pad (ptr_resultat^.res_com,' ',75);
                           stat := $put_screen(str_com,22,1,1);
                           affich_format(ptr_colonne);
                      end;
   if sais = -10 then afficher_next_com(ptr_commande);
 end;
 fin_intro := (sais = -1);
end;    (* of instring *)
```

```
(*      *       *       * procedure travail sur un res *       *       *)

procedure modifier_res (ptr_res : pointeur_res; ptr_lig : pointeur_ligne ;
                                tar,pro,ter : boolean; resultat : real);
var ptr_col : ^colonne;
    vrais : boolean;

begin
  ptr_res^.res_tar := tar;
  ptr_res^.res_pro := pro;
  ptr_res^.term := ter;
  ptr_res^.exist := true;
  ptr_res^.modif := true;
  ptr_res^.res_val := resultat;
(* verification demande effectuee *)
  if ptr_lig^.res_dout[curs.col] = false then ptr_res^.res_type := superflu
                                         else ptr_res^.res_type := ok;
(* verification resultat dans les normes *)
  acces_colonne(curs.col,ptr_col);
  valider_vraissembl (resultat,ptr_col^.res_min,ptr_col^.res_max,vrais);
  if (not vrais) then ptr_res^.res_type := anormal;
end; (* of modifier_res *)

procedure supprimer_res (ptr_res : pointeur_res; ptr_lig : pointeur_ligne );

begin
  buffer^.res_lig^.res_val := ptr_res^.res_val;   (* pour CB res *)
  ptr_res^.res_val := -1;
  ptr_res^.res_pro := true;     (* valeur par defaut *)
  ptr_res^.res_tar := true;     (* valeur par defaut *)
  ptr_res^.exist := false;
  ptr_res^.modif := true;
  ptr_res^.term := false;
  if (ptr_lig^.res_dout[curs.col] = true) then ptr_res^.res_type := manquant
                                          else ptr_res^.res_type := ok;
end;    (* of supprimer_res *)

procedure mod_add_sup_res (ch:char;num:integer;ptr_lig:pointeur_ligne;
                           ptr_res:pointeur_res;var stop :boolean);

var stat:integer;
    realstr : real;
    str,strbis,str_error:varying[80] of char;
    modif,correct,res_term,normal : boolean;
    lc : integer;
    rech_tab,tab : packed array[1..7] of char;


begin
  if ch <> 'i' then begin  ptr_commande := com_tab [9];
                           afficher_next_com(ptr_commande); end;
  str_error := '';
  set_pos_curs;
  acces_colonne(curs.col,ptr_colonne);
  affich_format(ptr_colonne);
  if ptr_res^.exist then convertir_ecrire (ptr_colonne^.format,
                                           ptr_res^.res_val,((not ptr_colonne^
                                               tar_col) or (ptr_res^.res_tar)
                           ptr_res^.res_pro,str)
                    else str:='';
  correct := false;
  while not correct do
    begin
        if ptr_res^.res_com <>'' then stat := $put_screen(ptr_res^.res_com,
                                                          22,1,1);
```

```
        correct := true;
        instring(str,num (*+ 1*),curs.lig - prem_lig_fe + deplig,curs.poscol(*
                 (ptr_colonne^.col_len - ptr_colonne^.res_len ) div 2 *),
                  ptr_colonne^.res_len,fill_char,0,modif,0,pwstop);
        if (length(str)<>0) then begin
                case str[1] of '.': begin stop:=true;
                                          correct:=true;
                                    end;

                               '\': begin
                                        supprimer_res(ptr_res,ptr_ligne);
                                        correct := true;
                                    end;

                               '&': begin
                    str_error := '';
                    strbis := pad(str,' ',6); strbis:=pad(strbis,'0',8);
                    str$upcase(strbis,strbis);
                    rech_tab := substr(strbis,2,length(strbis)-1);
                    repeat findk (fich_rem_std,0,rech_tab,GEQ,error:=continue);
                    until status(fich_rem_std) <= 0 ;
                    tab := fich_rem_std^.code_no_rem_std;
                    correct :=  not(ufb(fich_rem_std)) and (substr(tab,1,5)=
                                substr(rech_tab,1,5)) and  ((((tab[6]<>'0') or
                                 (tab[7]<>'0') and ptr_res^.com_long) or
                                    not ptr_res^.com_long));
                    if (correct) and (length(ptr_res^.res_com) + length(str) < 75)
                    then begin ptr_res^.res_com := ptr_res^.res_com + str;
                       str := '';
                       ptr_res^.com_long := (tab[6]<>'0') or (tab[7]<>'0');
                 end
                    else if ufb(fich_rem_std) or (substr(tab,1,5)<>substr(
                                rech_tab,1,5))then str_error := err_tab[12]
                                        else str_error := err_tab[13];
                    correct := false;
                end;
                               otherwise
                begin
                    valider_format(ptr_colonne^.format,str,
                    realstr,ptr_res^.res_tar,ptr_res^.res_pro,
                    ptr_res^.term,correct,normal,str_error);
                    modifier_res(ptr_res,ptr_ligne,ptr_res^.res_tar,
                                ptr_res^.res_pro,ptr_res^.term,realstr);
                 end;
                               end; (* case *)
                        end (* then *)
                        else    begin
                                    supprimer_res(ptr_res,ptr_ligne);
                                    correct := true;
                                end;
   end; (* of while *)
  affich_resultat;
  stat := $put_screen('           ',23,71,1);
  if ch <> 'i' then begin  ptr_commande := com_tab[1];
                           afficher_next_com(ptr_commande); end;
  if ptr_res^.res_com <>'' then stat := $put_screen(pad('',' ',80),22,1,1);
end;    (* of mod_add_sup_res *)
```

```
(*      *       *       *  procedure niveau superieures *       *       *)

procedure find_resultat (valeur:real);

var num_lig : integer;
    num_col : integer;
    trouve : boolean;

begin
  trouve := false;
  num_lig := curs.lig;
  num_col := curs.col;
  while (num_lig <= nbre_ligne_tableau) and (not trouve) do
  begin
        while (num_col <= nbre_colonne_tableau) and (not trouve) do
        begin
                trouve := (ptr_resultat^.res_val = valeur ) and
                                (ptr_resultat^.exist);
                num_col := num_col + 1;
                ptr_resultat := ptr_resultat^.next_res;
        end;
        if (not trouve) then num_col := prem_col;
        num_lig := num_lig + 1;
        ptr_ligne := ptr_ligne^.next_lig;
        ptr_resultat := ptr_ligne^.res_lig^.next_res;
  end;
  if trouve then begin curs.lig := num_lig-1;curs.col := num_col-1;end
            else display_error(err_tab[5],1);
  if (curs.lig > der_lig_fe) or (curs.lig < prem_lig_fe) or
     (curs.col > der_col_fe) or (curs.col < prem_col_fe) then
                calcul_fe_pt(curs.lig,curs.col);
  find_res(curs.lig,curs.col);
  calcul_pos_curs;
end;    (* of find_resultat *)
```

```
procedure intro_auto ;

var num_lig : integer;
    num_col : integer;
    stop : boolean;

begin
  ptr_commande := com_tab[2];
  afficher_next_com (ptr_commande);
  num_lig := curs.lig;
  acces_ligne(num_lig,ptr_ligne);
  ptr_ligne := ptr_ligne^.prev_lig;
  stop := false; pwstop := false ;
  sup_pos_curs;
  while (num_lig<=nbre_ligne_tableau) and (not stop) and (not pwstop) do
  begin
       ptr_ligne := ptr_ligne^.next_lig;
       num_col := prem_col;
       ptr_resultat := ptr_ligne^.res_lig;
       curs.lig := num_lig;
       while (num_col <= nbre_colonne_tableau) and (not stop) and
                              (not pwstop) do
       begin
              curs.col := num_col;
              ptr_resultat := ptr_resultat^.next_res;
              if (ptr_resultat^.res_type = manquant) and
                     (ptr_resultat^.res_com = '') then begin
                 if (curs.lig > der_lig_fe) or (curs.lig < prem_lig_fe) or
                        (curs.col > der_col_fe) or (curs.col < prem_col_fe)
                              then calcul_fe_pt_intro(curs.lig,curs.col);
                              calcul_pos_curs;
                              mod_add_sup_res ('i',1,ptr_ligne,
                                                  ptr_resultat,stop);
                              sup_pos_curs;
              end; (* IF *)
              num_col := num_col + 1;
       end;
       num_lig := num_lig + 1;
   end;
   find_res(curs.lig,curs.col);
   calcul_pos_curs;
   if (curs.lig > der_lig_fe) or (curs.lig < prem_lig_fe) or
      (curs.col > der_col_fe) or(curs.col < prem_col_fe)
    then calcul_fe_pt(curs.lig,curs.col);
   display_error(err_tab[9],1);
   ptr_commande := com_tab[1];
   afficher_next_com (ptr_commande);
end;    (* of intro_auto *)
```

```
(*      *       *       * procedure sur ligne de res *  *       *       *)

procedure supprimer_ligne_res (var ptr_lig : pointeur_ligne);

var ptr : ^res_ligne;
    temp : ^res_ligne;

begin
  ptr := buffer^.res_lig^.next_res;
  while ptr^.res_type <> premier do
  begin
        temp := ptr^.next_res;
        dispose (ptr);
        ptr := temp;
  end;
  dispose (ptr);
  buffer^.res_lig := ptr_lig^.res_lig;
end;    (* of supprimer_ligne_res *)


procedure ajouter_ligne_res ( var ptr_lig : pointeur_ligne );

var ptr : ^res_ligne;
    i : integer;
    temp : integer;

begin
  temp := curs.col;
  new(ptr);
  ptr^.next_res := ptr;
  ptr^.prev_res := ptr;
  ptr^.res_type := premier;
  ptr_lig^.res_lig := ptr;
  for i:=1 to nbre_colonne_tableau do begin creer_ptr_res (ptr);
                                            curs.col := i;
                                            supprimer_res(ptr,ptr_lig);
                                      end;
  curs.col := temp;
end;    (* of ajouter_ligne_res *)


procedure copy_buffer(ptr_lig : pointeur_ligne);

var temp:^ligne;
    i:integer;
    ptr_res,ptr_res_buffer : ^res_ligne;
    temp_col : integer;

begin
  temp := buffer;
  temp_col := curs.col;
  buffer := ptr_lig;
  supprimer_ligne_res(ptr_lig);
  ajouter_ligne_res(ptr_lig);
  buffer := temp;
  ptr_res := ptr_lig^.res_lig^.next_res;
  ptr_res_buffer := buffer^.res_lig^.next_res;
  for i:=1 to nbre_colonne_tableau do
  begin
        curs.col:=i;
        if ptr_res_buffer^.exist then modifier_res(ptr_res,ptr_lig,
                ptr_res_buffer^.res_tar,ptr_res_buffer^.res_pro,
                ptr_res_buffer^.term,ptr_res_buffer^.res_val)
                                  else supprimer_res (ptr_res,ptr_lig);
        ptr_res := ptr_res^.next_res;
```

```
  end;
  curs.col := temp_col;
  find_res(curs.lig,curs.col);
  affich_fenetre;
end;    (* of copy_buffer *)


procedure add_lig_res;

var ptr_lig : ^ligne;
    num_lig : integer;
    i,x     : integer;
    ptr_res : ^res_ligne;

begin
  x:=curs.col;
  ptr_lig := first_ligne^.prev_lig;
  supprimer_ligne_res (ptr_lig);
  num_lig := der_lig;
  while num_lig > curs.lig do
  begin
       ptr_lig^.num_godet := ptr_lig^.prev_lig^.num_godet;
       ptr_lig^.res_lig := ptr_lig^.prev_lig^.res_lig;
       ptr_res :=ptr_lig^.res_lig^.next_res;
       for i:= prem_col to der_col do
       begin
         curs.col := i;
         if ptr_res^.exist then modifier_res(ptr_res,ptr_lig,ptr_res^.res_tar
                               ,ptr_res^.res_pro,ptr_res^.term,
                                    ptr_res^.res_val)
                           else supprimer_res(ptr_res,ptr_lig);
         ptr_res := ptr_res^.next_res;
       end;
       ptr_lig := ptr_lig^.prev_lig;
       num_lig := num_lig - 1;
  end;
  ajouter_ligne_res (ptr_lig);
  ptr_lig^.num_godet := 0;
  curs.col := x;
end;    (* of add_lig_res *)


procedure sup_lig_res;

var ptr_lig : ^ligne;
    num_lig : integer;
    i,x     : integer;
    ptr_res : ^res_ligne;

begin
  x:=curs.col;
  acces_ligne(curs.lig,ptr_lig);
  supprimer_ligne_res (ptr_lig);
  num_lig := curs.lig;
  while num_lig < nbre_ligne_tableau do
  begin
       ptr_lig^.num_godet := ptr_lig^.next_lig^.num_godet;
       ptr_lig^.res_lig := ptr_lig^.next_lig^.res_lig;
       ptr_res :=ptr_lig^.res_lig^.next_res;
       for i:= prem_col to der_col do
       begin
         curs.col := i;
         if ptr_res^.exist then modifier_res(ptr_res,ptr_lig,ptr_res^.res_tar
                                             ptr_res^.res_pro,ptr_res^.term
                                             ptr_res^.res_val)
```

```
          ptr_res := ptr_res^.next_res;
        end;
        ptr_lig := ptr_lig^.next_lig;
        num_lig := num_lig + 1;
  end;
  ajouter_ligne_res (ptr_lig);
  ptr_lig^.num_godet :=0;
  curs.col := x;
end;    (* of sup_lig_res *)


procedure add_sup_lig_res;

var ch:char;
    chspec : spec_char;

begin
 ptr_commande := com_tab[4];
 afficher_next_com (ptr_commande);
 lec_touche(ch,chspec);
 case ch of '+' : add_lig_res;
            '-' : sup_lig_res;
        end;
 find_res(curs.lig,curs.col);
 affich_fenetre;
end;    (* of add_sup_lig_res *)
```

```
procedure rech_val;

var noerror,modif : boolean;
    i,x,y : integer;
    str : varying[10] of char;

function valider_reel ( str : varying[c] of char; var val:real) : boolean ;
(* reel positif : ? tester status *)
var i : integer;

begin
 val := -1;
 readv(str,val,error:=continue);
 valider_reel := (val <> -1);
end;    (* of valider_reel *)

begin
  ptr_commande := com_tab[5];
  afficher_next_com (ptr_commande);
  sup_pos_curs;
  str := '';
  instring(str,1,23,70,8,'*',1,modif,0,pwstop);
  noerror := valider_reel ( str ,pwreel );
  if noerror then  find_resultat(pwreel)
             else display_error(err_tab[11],1);
  ptr_commande := com_tab[1];
  afficher_next_com (ptr_commande);
  i := $put_screen('          ',23,70,1);
end;    (* of rech_val *)
```

```
procedure rech_ident;

var str:varying[10] of char;
    tab : packed array[1..6] of char;
    modif,noerror : boolean;
    i : integer;

function valider_int (str : varying[c] of char ; var val : integer) : boolean
(* int >= 0 *)

var i : integer;

begin
 val := -1;
 readv(str,val,error:=continue);
 valider_int := (val <> -1);
end;    (* of valider_int *)

begin
  str := '';
  ptr_commande := com_tab[6];
  afficher_next_com (ptr_commande);
  sup_pos_curs;
  instring(str,1,23,70,6,'*',1,modif,0,pwstop);
  noerror := valider_int ( str , i );
  if noerror then
  begin
    str:=pad('','0',6-length(str)) + str;
    for i := 1 to 6 do tab[i] := str [i];
    i := prem_lig;
    trouve := false;
    ptr_lig := first_ligne;
    while (i<=nbre_ligne_tableau) and (not trouve) do
    begin
     ptr_lig := ptr_lig^.next_lig;
     trouve := (ptr_lig^.ident = tab);
     i:=i+1;
    end;
    if trouve then
              begin find_res(i-1,prem_col);
                    curs.lig := i-1;
                    curs.col := prem_col;
              end
            else display_error(err_tab[6],1);
  end
  else display_error(err_tab[11],1);
  calcul_fe_pt (curs.lig,curs.col);
  calcul_pos_curs;
  ptr_commande := com_tab[1];
  afficher_next_com (ptr_commande);
  i := $put_screen('          ',23,70,1);
end;    (* of rech_ident *)
```

```
procedure jump;

var error,modif : boolean;
    i,x,y : integer;
    str : varying[10] of char;

function valider_int ( str : varying[c] of char; var val : integer) : boolean

var i : integer;

begin
 readv(str,val,error:=continue);
 valider_int := (val <> -1);;
end;    (* of valider_int *)

begin
  x:= curs.lig; y:=curs.col;
  str := '';
  sup_pos_curs;
  ptr_commande := com_tab[7];
  afficher_next_com (ptr_commande);
  instring(str,1,23,70,6,'*',1,modif,0,pwstop);
  error := valider_int ( str , x );
  str := '';
  afficher_next_com (ptr_commande);
  instring(str,1,23,70,6,'*',1,modif,0,pwstop);
  error := valider_int ( str , y);
  if x> der_lig then x:=der_lig;
  if x< prem_lig then x := prem_lig;
  if y> der_col then y:=der_col;
  if y< prem_col then y := prem_col;
  curs.lig :=x; curs.col := y;
  find_res(curs.lig,curs.col);
  calcul_fe_pt(curs.lig,curs.col);
  calcul_pos_curs;
  ptr_commande := com_tab[1];
  afficher_next_com (ptr_commande);
  i := $put_screen('          ',23,70,1);
end;    (* of jump *)

procedure fenetre_commentaire;

var pt_trav : point_com;
    stat : integer;
    str_res : varying[80] of char;

procedure transfert_str_p_com (str : varying[c] of char);

var ptr:point_com;

begin
  new (ptr);
  ptr^.next := nil;
  ptr^.prec := nil;
  ptr^.texte := str40(str);
  init_com := ptr;
  while length(str) > 0 do
  begin
        creer_p_com(ptr);
        ptr^.texte := str40(str);
  end;
end;    (* of transfert_str_p_com *)

procedure transfert_p_com_str (var str : varying[c] of char);
```

```
  stat := $put_screen(pad('',' ',80),23,1,0);

  if init_com <> nil then
  begin
    ptr_resultat^.modif := true;
    transfert_p_com_str(str_res);
    if str_res = 'œœœœœœ' then begin ptr_resultat^.res_com := pad(
'IMPOSSIBLE DE CHARGER TOUT LE COMMENTAIRE ... PASSEZ PAR LA FENETRE',' ',75);
                                    ptr_resultat^.com_too_long := true;
                                    decharger_com;
                               end
                          else begin
                                 ptr_resultat^.res_com := str_res;
                                 ptr_resultat^.com_long := long(ptr_resultat);
                                 ptr_resultat^.com_too_long := false;
                               end;
  end;
  sup_p_com;
end;    (* of fenetre_commentaire *)
```

```
(*      *       *       * procedure initialisation *    *       *       *)

procedure lecture_fich_err;

var i : integer;
    fich_err : text;

begin
   for i:=1 to nbre_erreur do err_tab[i] := 'undefined error';
   open (fich_err,'vl_online:erreur.dta',readonly,error := continue);
   reset(fich_err,error := continue);
   i:=1;
   while (i<=nbre_erreur) and (status(fich_err)=0) do
   begin
        readln(fich_err,err_tab[i]);
        i := i + 1;
   end;
   close (fich_err,error := continue);
end;    (* of lecture_fich_err *)


procedure lecture_fich_com;

const long = 70;

var fich_com : text;
        i:integer;
        str:varying[long] of char;

begin
  for i:=0 to nbre_fonction do
  begin
        new(com_tab[i]);
        com_tab[i]^.str := pad('NO COMMENT',' ',long);
        com_tab[i]^.next_com := com_tab[i];
        com_tab[i]^.prev_com := com_tab[i];
  end;
  open(fich_com,'vl_online:commande.dta',readonly,sharing:=readonly,
                        error := continue);
  reset(fich_com,error := continue);
  i:=1;
  while (status(fich_com)=0) and (i<=nbre_fonction) do
  begin
        readln(fich_com,str,error := continue);
        ptr_commande := com_tab[i];
        while (str<>'end') and (str<>'END') and (not eof(fich_com)) do
        begin
                creer_ptr_com(ptr_commande);
                ptr_commande^.str:=pad(str,' ',long);
                readln(fich_com,str,error := continue);
        end;
        i:=i+1;
  end;
  close (fich_com,error := continue);
end;    (* of lecture_fich_com *)
```

```
procedure lecture_colonne (num_JBL : integer);

var (*fich_ana_res : file of rec_sig_ana;
    fich_sig_ent_jbl : file of rec_sig_ent_jbl;
    fich_sig_ana_jbl : file of rec_sig_ana_jbl;*)
    buf,temp : packed array[1..4] of char;
    i : integer;
    ptr_col : pointeur_colonne;

begin
   findk(fich_sig_ent_jbl,0,num_jbl,error := continue);
   if status(fich_sig_ent_jbl) <> 0 then error_sys_stop;
   titre_jbl := fich_sig_ent_jbl^.sig_tit_ent_jbl;
   nbre_colonne_tableau := fich_sig_ent_jbl^.sig_nb_ana_ent_jbl;
   ptr_col := first_colonne;
   buf[1] := chr (num_jbl div 10 + 48);
   buf[2] := chr (num_jbl mod 10 + 48);
   for i:=1 to nbre_colonne_tableau do
   begin
        buf[3] := chr (i div 10 + 48);
        buf[4] := chr (i mod 10 + 48);
        creer_ptr_col( ptr_col);
        findk (fich_sig_ana_jbl,0,buf,error := continue);
        if status(fich_sig_ana_jbl) <> 0 then error_sys_stop;
          ptr_col^.nom := fich_sig_ana_jbl^.sig_code_ana_jbl;
          ptr_col^.col_len := fich_sig_ana_jbl^.sig_larg_ecr_ana_jbl + 1;
          ptr_col^.num_col := i;

        findk (fich_sig_ana,0,ptr_col^.nom,error := continue);
        if status(fich_sig_ana) <> 0 then error_sys_stop;
          ptr_col^.res_min := fich_sig_ana^.sig_val_vrais_inf_ana;
          ptr_col^.res_max := fich_sig_ana^.sig_val_vrais_sup_ana;
          ptr_col^.format  := fich_sig_ana^.sig_typ_fmt_res_ana;
          ptr_col^.tar_col := fich_sig_ana^.sig_ind_ana_tarif_ana;
          case ptr_col^.format[1] of
        'D','I','d','i':ptr_col^.res_len := ord (ptr_col^.format[2]) - 46;
                'F','f':ptr_col^.res_len := ord (ptr_col^.format[2]) - 44;
                'R','r':ptr_col^.res_len := ord (ptr_col^.format[2]) - 48 + 1 -
                                            ord (ptr_col^.format[4]) - 48 + 2;
                'A','a':ptr_col^.res_len := ord (ptr_col^.format[2]) - 48 + 2;
        'S','G','s','g':ptr_col^.res_len := 4;
                'B','b':ptr_col^.res_len := 5;
                'P','p':ptr_col^.res_len := 6;
                'C','c':ptr_col^.res_len := 7;

          end;  (* case *)
          if ptr_col^.res_len < 4 then ptr_col^.res_len := 4;

   end;

end;    (* of lecture_colonne *)
```

```
procedure init_tableau(num_jbl:integer);

type prim = record case boolean of
                true :(jbl_ref_pl : packed array[1..10] of char);
                false :(jbl : packed array[1..2] of char;
                        ref : packed array[1..6] of char;
                        pl  : packed array[1..2] of char);
        end;

var clef : prim;
    str : varying[11] of char;
    ptr_lig :^ligne;
    ptr_col :^colonne;
    ptr_res :^res_ligne;
    tab : packed array[1..11] of char;
(*    fich_rec_jbl : file of rec_jbl;
    fich_rec_ana_res : file of rec_ana_res; *)
    temp_jbl : packed array[1..2] of char;
    temp_ref : packed array[1..6] of char;
    i,j,num,x,z,y : integer;
    num_pl,tentative,num_edition : integer;
    stop,okopen : boolean;

procedure lecture_commentaire (ptr_res : pointeur_res);

var continue : boolean;
    cont : integer;

begin
  continue := true;
  ptr_res^.res_com := '';
  cont := fich_ana_res^.pt_res_com;
  while (cont<>0) and (continue) do
  begin
        repeat find(fich_ext_res_ana,cont,error:=continue);
        until (status(fich_ext_res_ana)<>74);
        if length(ptr_res^.res_com) +
                length(fich_ext_res_ana^.txt_ext_res_ana) < 75
        then ptr_res^.res_com := ptr_res^.res_com +
                        fich_ext_res_ana^.txt_ext_res_ana + 'œ'
        else begin ptr_res^.res_com :=
 'IMPOSSIBLE DE CHARGER TOUT LE COMMENTAIRE ... PASSEZ PAR LA FENETRE';
                   ptr_res^.res_com := pad(ptr_res^.res_com,' ',75);
                   ptr_res^.com_too_long := true;
                   continue := false;
             end;
        cont := fich_ext_res_ana^.pt_next_ext_res_ana;
  end;
  ptr_res^.com_long := long(ptr_res);
end;    (* of lecture_commentaire *)

begin
  ptr_lig := first_ligne;
  clef.jbl_ref_pl := '00' + prem_ref + '00';
  clef.jbl[1] := chr(num_jbl div 10 + 48);
  clef.jbl[2] := chr(num_jbl mod 10 + 48);
  temp_jbl := clef.jbl;
  repeat findk(fich_jbl,0,clef.jbl_ref_pl,geq,error := continue);
  until status(fich_jbl)<=0;
  stop := false;
  i:=0;
  while (not ufb(fich_jbl)) and (not stop) do
  begin
        clef.jbl_ref_pl := fich_jbl^.no_jbl_no_ref_pl_jbl;
        if (temp_jbl <> clef.jbl) or (clef.ref > dern_ref) then stop := true
```

```
                if temp_ref <> clef.ref then
                begin
                        i:=i+1;
                        creer_ptr_lig(ptr_lig);
                        ptr_lig^.ident := clef.ref;
                        temp_ref := clef.ref;
                        stat := $put_screen(clef.ref,22,1,1);
                        ptr_res := ptr_lig^.res_lig;
                        for num:=1 to nbre_colonne_tableau do begin
                                                creer_ptr_res(ptr_res);
                                                ptr_lig^.res_dout[num] := fals
                                                        end;
                end;
                writev(str,clef.pl);
                readv(str,num_pl);
                ptr_lig^.res_dout[num_pl] := true;
                ptr_res := ptr_lig^.res_lig;
                acces_resultat (num_pl,ptr_res);
                ptr_res^.res_type := manquant;
                curs.col := num_pl;
                repeat findk(fich_ana_res,0,fich_jbl^.pt_res_jbl,
                        error:=continue) until status(fich_ana_res) <=0;
                if (fich_ana_res^.res_term)  and ((fich_ana_res^.res_num_ana
                        <> -1) or (fich_ana_res^.pt_res_com <> 0) or
                   (not fich_ana_res^.ind_ana_tarifiable_res) or
                        (not fich_ana_res^.ind_ana_protocolable_res ))
                  then  (* kaiiiiii *)
                  modifier_res(ptr_res,ptr_lig,
                                fich_ana_res^.ind_ana_tarifiable_res,
                                fich_ana_res^.ind_ana_protocolable_res,
                                fich_ana_res^.res_term,
                                fich_ana_res^.res_num_ana)
                else
                supprimer_res(ptr_res,ptr_lig);
                ptr_res^.modif := false;
                lecture_commentaire(ptr_res);

                                end;
                tentative := 0;
                repeat
                  get(fich_jbl,error := continue);
                  okopen := (status(fich_jbl) <= 0);
                  tentative := tentative + 1;
                  if tentative >= nbre_tentative_error then
                        begin error_sys_more;
                              tentative := 0;
                        end;
                until okopen;
        end; (* while *)
  unlock(fich_jbl,error := continue);
  unlock(fich_ana_res,error:=continue);
  unlock(fich_ext_res_ana,error:=continue);
  nbre_ligne_tableau := i;
  stat := $put_screen('                ',22,1,1);
end;    (* of init_tab *)
```

```
procedure heure_date;   (* certainement possible de BEAUCOUP MIEUX mais ??? *)

var str:packed array[1..11] of char;

begin
   time(str);
   heure_sys[1] := str[1];
   heure_sys[2] := str[2];
   heure_sys[3] := str[4];
   heure_sys[4] := str[5];
   if heure_sys[1] = ' ' then heure_sys[1] := '0';
   if heure_sys[3] = ' ' then heure_sys[3] := '0';

   date (str);
   date_sys[5] := str[1];
   date_sys[6] := str[2];
   date_sys[1] := str[10];
   date_sys[2] := str[11];
   str := substr('010203040506070809101112',(index('
                        JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC',substr(str,4,3))
                                DIV 3)*2+1,2);
   date_sys[3] := str[1];
   date_sys[4] := str[2];
   if date_sys[5]=' ' then date_sys[5]:='0';
end;    (* of heure_date *)
```

```
procedure init;

var ch:char;
    chspec : spec_char;
    stat,i,j : integer;
    ptr_col : ^colonne;
    ptr_lig : ^ligne;
    ptr_res : ^res_ligne;

begin
(* ****************************************** *)
(* initialisation necessaire pour SMG     *)
  smg$create_pasteboard(pasteboard_id := pasteboard_1);
  smg$create_virtual_display(rows:= 24,columns := 80,
       display_id := displgen);
  smg$paste_virtual_display(displgen,pasteboard_1,1,1);
  smg$put_chars(displgen,' ',1,1,,0);

(* ****************************************** *)
(* initialisation des commentaires et erreur*)

  lecture_fich_com;
  lecture_fich_err;

(* ****************************************** *)
(* fenetre d'attente *)

stat := $erase_page(1,1);
entete;
ptr_commande := com_tab[8];
afficher_next_com(ptr_commande);

(* ****************************************** *)
(* initialisation des codes speciaux de res *)

  init_table;

(* ****************************************** *)
(* initialisation de UC_tab *)

  chspec:=UC_REC;
  for i:=1 to 49 do begin UC_tab[i]:=chspec;
                          chspec:=succ(chspec);
                    end;
  UC_tab[50]:=chspec;

(* ****************************************** *)
(* init du channel pour qiow *)

  stat := $assign ('sys$input',dev_chan);

(* ****************************************** *)
(* creation des premiers elements du tableau *)

  new(first_ligne);
  first_ligne^.next_lig:=first_ligne;
  first_ligne^.prev_lig:=first_ligne;
  first_ligne^.num_lig := '000000';
  new(first_ligne^.res_lig);
  first_ligne^.res_lig^.next_res:=first_ligne^.res_lig;
  first_ligne^.res_lig^.prev_res:=first_ligne^.res_lig;
  ptr_resultat := first_ligne^.res_lig;
  ptr_resultat^.res_type := premier;
  for i:= 1 to nbre_colonne_tableau do creer_ptr_res(ptr_resultat);
```

```
  first_colonne^.next_col:=first_colonne;
  first_colonne^.prev_col:=first_colonne;

(* ****************************************** *)
(* creation du buffer *)

  new(buffer);
  new(buffer^.res_lig);

        ptr_res:=buffer^.res_lig;
        ptr_res^.res_type := premier;
        ptr_res^.next_res := ptr_res;
        ptr_res^.prev_res := ptr_res;
        for j:=1 to nbre_colonne_tableau do creer_ptr_res(ptr_res);

(* ****************************************** *)
(* initialisation des colonnes du tableau *)
  nbre_ligne_tableau := 0;
  if num_jbl <>0 then lecture_colonne ( num_jbl );

(* ****************************************** *)
(* initialisation du tableau *)

  if num_jbl<>0 then  init_tableau( num_jbl );

(* ****************************************** *)
(* mise a jour des variables globales *)

  ptr_lig := first_ligne;
  high := false;
  if nbre_ligne_tableau = 0 then begin
                                      fin_pgm := true;
                                      display_error(err_tab[7],1);
                                      lec_touche(ch,chspec);
                                end
                           else fin_pgm := false;
  heure_date;
  prem_lig := 1;
  prem_col := 1;
  der_lig := nbre_ligne_tableau;
  der_col := nbre_colonne_tableau;
  prem_lig_fe := 0;
  der_lig_fe := 0;
  prem_col_fe := 0;
  der_col_fe := 0;
  nbre_lig_fe := 10;
  nbre_col_fe := 80 - depcol;
  curs.lig := 1;
  curs.col := 1;
  curs.poscol := depcol;
  ptr_ligne := first_ligne;
  ptr_colonne := first_colonne;
  ptr_resultat := first_ligne^.next_lig^.res_lig;
  ptr_commande := com_tab[1];

(* ****************************************** *)
(* presentation des resultats *)

  display_titre ('INTRODUCTION  ' + titre_jbl);
  calcul_fe_pt (curs.lig,curs.col);
  ptr_commande := com_tab[1];
  afficher_next_com (ptr_commande);
  set_pos_curs;

(* ****************************************** *)
```

```
end; (* of init *)
```

```
(*      *       *       * procedure terminaison *       *       *       *)

procedure term(num_jbl:integer);

var (* fich_rec_ana_res : file of rec_ana_res; *)
    num_lig : integer;
    num_col,tentative : integer;
    continue,okopen : boolean;
    z : integer;

procedure sauver_commentaire(ptr_res : pointeur_res);

var ptr,ptr_temp,ptr_temp2:integer;
    str,strtemp : varying[80] of char;
    temp : rec_ext_res_ana;

begin
  if ptr_res^.com_too_long = false then
  begin
    str := ptr_res^.res_com;
    if length(str) = 0 then begin
                                ptr_temp := fich_ana_res^.pt_res_com;
                                fich_ana_res^.pt_res_com := 0;
                                while ptr_temp <>0 do
                                begin
          find(fich_ext_res_ana,ptr_temp);
          ptr_temp2 := fich_ext_res_ana^.pt_next_ext_res_ana;
          rel_rec_ext_res_ana(ptr_temp);
          ptr_temp := ptr_temp2;
                                end;
                            end
                       else begin
       if fich_ana_res^.pt_res_com = 0
              then begin new_rec_ext_res_ana(ptr);
                                fich_ana_res^.pt_res_com := ptr;end
              else ptr := fich_ana_res^.pt_res_com;
       temp.txt_ext_res_ana := str40(str);
       while length(str)>0 do
          begin
              find(fich_ext_res_ana,ptr);
              if fich_ext_res_ana^.pt_next_ext_res_ana = 0
                      then new_rec_ext_res_ana(ptr_temp)
                      else ptr_temp :=fich_ext_res_ana^.pt_next_ext_res_ana;
              temp.pt_next_ext_res_ana := ptr_temp;
              find(fich_ext_res_ana,ptr);
              fich_ext_res_ana^.pt_next_ext_res_ana :=
                      temp.pt_next_ext_res_ana;
              fich_ext_res_ana^.txt_ext_res_ana := temp.txt_ext_res_ana;
              update (fich_ext_res_ana);
              ptr:= ptr_temp;
              temp.txt_ext_res_ana := str40(str);
           end;
           temp.pt_next_ext_res_ana := 0;
           find(fich_ext_res_ana,ptr);
           ptr_temp := fich_ext_res_ana^.pt_next_ext_res_ana;
           while ptr_temp <>0 do
           begin
              find(fich_ext_res_ana,ptr_temp);
              ptr_temp2 := fich_ext_res_ana^.pt_next_ext_res_ana;
              rel_rec_ext_res_ana(ptr_temp);
              ptr_temp := ptr_temp2;
           end;
           find (fich_ext_res_ana,ptr);
           fich_ext_res_ana^.pt_next_ext_res_ana := temp.pt_next_ext_res_ana;
          fich_ext_res_ana^.txt_ext_res_ana := temp.txt_ext_res_ana;
```

```
                        end; (* of else *)
  end; (* if *)
  unlock(fich_ext_res_ana,error:=continue);
end;    (* of sauver_commentaire *)

procedure sauver_ligne_bd (ptr_lig : pointeur_ligne );

var x,i : integer;
    ptr_res : pointeur_res;
    ptr_col : pointeur_colonne;
    str : varying [11] of char;
    tab : packed array [1..11] of char;
    okopen : boolean;
    stat,tentative : integer;

begin
  ptr_res := ptr_lig^.res_lig;
  stat := $put_screen(ptr_lig^.ident,22,1,1);
  for i:= 1 to nbre_colonne_tableau do
  begin
        ptr_res := ptr_res^.next_res;
        if (ptr_res^.modif) then
        begin
                acces_colonne (i,ptr_col);
                writev(str,ptr_lig^.ident:6,ptr_col^.nom:5);
                tab := str ;
                tentative := 0;
                repeat
                  findk(fich_ana_res,1,tab,error:=continue);
                  okopen := (status(fich_ana_res) <= 0);
                  tentative := tentative + 1;
                  if tentative >= nbre_tentative_error then
                                begin error_sys_more;
                                      tentative := 0;
                                end;
                until okopen;

                sauver_commentaire(ptr_res);
                fich_ana_res^.ind_ana_tarifiable_res := ptr_res^.res_tar and
                                ptr_col^.tar_col;
                fich_ana_res^.ind_ana_protocolable_res := ptr_res^.res_pro;
                fich_ana_res^.res_term := ((ptr_res^.term) or
                                (fich_ana_res^.pt_res_com<>0));
                fich_ana_res^.date_intro_res := date_sys;
                fich_ana_res^.heure_intro_res := heure_sys;
                fich_ana_res^.date_real_ana := date_sys;
                fich_ana_res^.heure_real_ana := heure_sys;
                fich_ana_res^.res_num_ana := ptr_res^.res_val;
                fich_ana_res^.res_prot := false;
                if not ufb(fich_ana_res) then update (fich_ana_res,
                                                      error:=continue);
                (* si ufb alors fichier journal mis a jour.
                        cfr ROULIN      *)
        end;
  end;
end;    (* sauver_ligne_bd *)

begin
  ptr_commande := com_tab[11];
  afficher_next_com(ptr_commande);
  num_lig := prem_lig;
  ptr_ligne := first_ligne;
  for z := prem_lig to der_lig do begin
                                  ptr_ligne := ptr_ligne^.next_lig;
                                  sauver_ligne_bd (ptr_ligne);
```

```
    fin_pgm := true;
    suppr_ptr_col ( first_colonne);
    suppr_ptr_lig ( first_ligne);
end; (* of term *)
```

```
(*      *       *       *       programme       *       *       *       *)

begin
        readv(tab_num_jbl,num_jbl);
        init;
        while (not fin_pgm) do
        begin
                lec_touche(ch,chspec);
                erase_error;
                set_pos_curs;
                case chspec of
                UC_BS,UC_LF,    (* because of shit rainbow *)
                UC_F12,UC_F13,UC_PF2,UC_PF3,
                UC_INS,UC_SEL,UC_PUP,UC_PDO,
                UC_CUP,UC_CDO,UC_CLE,UC_CRI :begin
                                                deplacement(chspec,high);
                                                high:=false;
                                             end;
                UC_TAU : afficher_next_com(ptr_commande);
                UC_TIL : high := not high;
         UC_f11,UC_ESC : if ptr_ligne^.res_dout[curs.col]
                 then begin fenetre_commentaire;afficher_next_com
                                (ptr_commande);affich_resultat;end
                 else display_error(err_tab[14],1);

                UC_PF1 : begin
                                ptr_commande := com_tab[10];
                                chspec := UC_TAU;
                                while chspec=UC_TAU do
                                begin
                                afficher_next_com(ptr_commande);
                                lec_touche(ch,chspec);
                                end;
                                case ch of
                                        'e','E':term( num_jbl );
                                        'f','F':rech_val;
                                        'I','i':intro_auto;
                                        'v','V':(* correct_auto *);
                                        'c','C':(* commentaire *);
                                        'r','R':add_sup_lig_res;
                                        'q','Q':fin_pgm:=true;
                                        'n','N':rech_ident;
                                        '?'    :jump;
                                        'b','B':copy_buffer(ptr_ligne);
                                end;
                                ptr_commande := com_tab[1];
                                afficher_next_com(ptr_commande);
                         end;
                otherwise  if ptr_ligne^.res_dout[curs.col]
                        then mod_add_sup_res(ch,1,ptr_ligne,ptr_resultat,stop)
                        else display_error(err_tab[14],1);
                end;
                set_pos_curs;
        end;
   pwstat := $erase_page(1,1);
   9999 : pwstat := $erase_page(24,79);
end;    (* of pwscroll *)

end.
```

EPURATION MANUELLE

Texte des instructions pascal du programme d'épuration.

```
program ess (input,output);

var x,y,z : integer;

procedure epuration_online (JBL : integer; prem_ident, dern_ident : integer);

type
        tab6 = packed array[1..6] of char;

        res_temp = record
                rt_jbl_lig_pos :[key(0)] packed array[1..10] of char;
                rt_res : real;
                rt_num_JBL : integer;
                rt_num_col : integer;
                rt_num_lig : integer;
                rt_num_pos : integer;
                rt_tar : boolean;
                rt_pro : boolean;
                rt_acc : boolean;
        end;

var inf,outf : text;
    fich_res : file of res_temp;
    okopen,trouve,stop : boolean;
    str,rem : varying[80] of char;
    num_lig_old,num_lig,num_ident : integer;
    ident_JBL,ident : packed array[1..2] of char;

procedure supprimer_res(lig:integer);

type variant = record case boolean of
        true : (all:packed array[1..10] of char);
        false : (jbl : packed array[1..2] of char;
                 lig : packed array[1..6] of char;
                 pos : packed array[1..2] of char);
        end;

var temp : variant;
    tab : packed array[1..10] of char;
    num_lig : tab6;
    num_jbl : packed array[1..2] of char;

begin
  writev(str,lig:1);
  str := pad('','0',6-length(str)) + str;
  readv(str,num_lig);
  writev(str,JBL:1);
  str := pad('','0',2-length(str)) + str;
  readv(str,num_JBL);

  open( fich_res,'vl_online:res_temp',unknown,sharing:=readwrite,
        organization := indexed, access_method := keyed,error := continue);
  resetk(fich_res,0,error:=continue);
  tab := '0000000000';
  tab[1] :=chr( JBL div 10 + 48);
  tab[2] :=chr( JBL mod 10 + 48);
  findk(fich_res,0,tab,GEQ,error := continue);

  temp.all := fich_res^.rt_JBL_lig_pos;
  while (status(fich_res)=0) and (temp.lig <= num_lig) and
                (temp.jbl = num_jbl) do
  begin
        delete(fich_res,error:=continue);
        get(fich_res,error:=continue);
        temp.all := fich_res^.rt_JBL_lig_pos;
```

```
end;    (* of supprimer_res *)

begin
  okopen := false;
  while not okopen do
  begin
        open(inf,'vl_online:param_online',unknown,sharing := none,
                        error := continue);
        okopen := (status(inf) = 0);
        if not okopen then close(inf);
  end;
  open (outf,'vl_online:param_online',new);
  reset(inf,error := continue);
  rewrite (outf,error := continue);

  trouve := false;
  while (status(inf)=0) and (not trouve) do
  begin
        readln(inf,str);
        readv(str,ident_JBL,num_ident,num_lig,rem,error:=continue);
        if (rem[1] = '*') and (num_lig = JBL) then trouve := true;
        writeln(outf,str);
  end;

  num_lig_old := 0;
  ident_JBL := 'sy';
  stop := false;
  while (not stop) and (status(inf)<>-1) do
  begin
        readln(inf,str,error:=continue);
        readv(str,ident,num_ident,num_lig,rem,error := continue);
        stop := (ident <> ident_JBL) or (num_ident >= dern_ident);
        if (not stop) or ((num_ident>=prem_ident) and (num_ident<=dern_ident))
                then num_lig_old := num_lig;
  end;

  if (num_ident>dern_ident) or (ident<>ident_JBL) then writeln(outf,str);
  if ((ident <> 'sy') and (prem_ident>num_ident)) or (status(inf)=-1)
                then num_lig_old := 999999;
  supprimer_res(num_lig_old);

  while status(inf)<>-1 do
  begin
        readln(inf,str,error:=continue);
        writeln(outf,str,error:=continue);
  end;

  close(inf,delete,error:=continue);
  close(outf,error := continue);

end;    (* of epuration_ONLINE *)

begin
  writeln('jbl , prem,  dern numero d ident ');
  readln(x);
  readln(y);
  readln(z);
  epuration_online(x,y,z);
end.
```

MODE D'EMPLOI DU PROGRAMME DE VALIDATION

VAX-LAB

MODE D'EMPLOI DU LOGICIEL D'ACCEPTATION DES DONNEES TRANSMISES PAR ON-LINE

P.Wautié.

MARS 1987

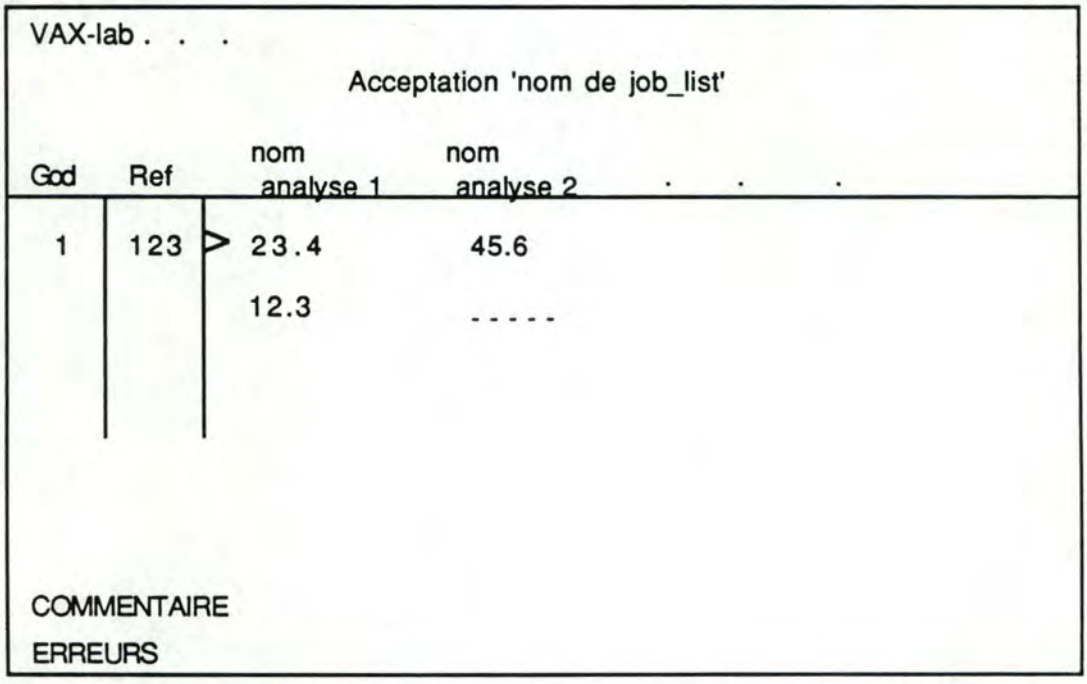
MODE D'EMPLOI DE 'VALIDATION'

But du programme

Dans l'utilisation d'un ONLINE, des problèmes de transfert peuvent apparaître. Ce programme permet une vérification (et une correction si nécessaire) avant de placer ces résultats dans la base de données.

Etudes des différentes phases du programme

Le programme est lancé via un système de choix multiples. Un premier écran vous est proposé ainsi qu'un message vous demandant de patienter quelques secondes. Il est à remarquer que l'attente peut durer assez longtemps. Patience donc ... L'initialisation terminée, l'écran se complète et affiche une partie des résultats transmis. Je vous présente ci-après le premier écran en vous précisant toutes les caractéristiques de celui-ci.



a) Titre général

Titre commun à toutes les fonctions du programme 'LABO'.

b) Titre du programme

Ce titre est le nom de la fonction dans ce programme 'LABO'. Il est composé de deux choses:

- du terme acceptation indiquant qu'on se trouve bien dans le programme d'acceptation du ONLINE.
- du titre de la JBL concernée par cette acceptation. Actuellement ce programme ne peut être utilisé que pour deux JBL différentes : Hitachi et Coulter (d'autres possibilités sont à l'étude : Cobas, BNA).

c) La ligne suivante est composée de signes dont voici la signification

'god' : rapelle que la colonne est réservée aux numéro de godet utilisé pour ces analyses.

'ident' : donne la colonne des numéros identifiants des demandes d'analyses. (000000 - 999999)

les noms d'analyses : cinq lettres maximum donnent le code de l'analyse. Ces codes sont définis par ailleurs (sigana.dat).

d) Le tableau

Les résultats des machines d'analyses sont dans la matrice ayant pour abscisses les noms des analyses et pour ordonnées les numéros d'identifiants ainsi que les numéros de godets associés. Les résultats sont de quatre types différents

- anormaux : les résultats sont en-dehors des normes de vraissemblance consacrées à cette analyse pour des sujets sains. (Ils apparaissent en gras à l'écran)

- superflus : les résultats sont transmis par la machine bien qu'ils n'avaient pas été programmés par le secrétariat. Ces résultats peuvent ne correspondre à aucune demande. (Ils apparaissent en clignotant à l'écran)

- manquants : ce sont les résultats programmés par le secrétariat mais qui n'ont pas été transmis par le ONLINE. (ceux-ci sont représentés par des barres horizontales)

- non-douteux : ce sont les résultats qui semblent être corrects (ni anormaux, ni superflus, ni manquants)

e) Le curseur

Ce curseur est représenté par un '´' clignotant. Ce curseur donne la position courante, c'est-à-dire donne le résultat, la ligne ou la colonne sur lesquels des modifications peuvent être effectuées.

f) La ligne de commandes

Cette ligne reprend le nom de la fonction dans laquelle on se trouve ainsi que toutes les possibilités liées à celle-ci. Si les possibilités ne tiennent pas en une seule ligne, il est possible de voir les autres en poussant sur `tab´. Si le fichier de commandes est inaccessible, la ligne est remplacée par 'no comment'

g) Ligne d'erreur et d'avertissement

Celle-ci est la dernière de l'écran et reprend toutes les erreurs et possibilités d'avertissement. Si le fichier des erreurs n'est pas accessible, la ligne 'undefined error' est présentée en cas d'erreur.

Première possibilité : le déplacement

Voilà vous vous trouvez dans un programme en état de marche (aucun problème n'est apparu lors de l'initialisation). La première possibilité qui vous est offerte est le déplacement du curseur ou des fenêtres. Vous pouvez voir sur la ligne de commandes, l'ensemble des possibilités qui existent dans ce niveau. Examinons-les en détail.

Le déplacement curseur.

`les flèches´ : celles-ci se trouvent sur la droite du clavier alphabétique et sur la gauche du bloc numérique. En poussant sur l'une de ces flèches, le curseur se déplace d'un résultat dans la direction de la flèche. Si vous vous trouvez sur un bord du tableau, et si vous cherchez à vous déplacer vers l'extérieur de celui-ci, un message d'erreur vous est présenté et le curseur ne quitte pas sa position d'origine. Si vous vous trouvez sur le bord d'une fenêtre sans être sur le bord du tableau, un déplacement curseur donne lieu à la présentation d'une nouvelle fenêtre et à un positionnement sur l'élément demandé (flèches). Les fenêtres sont calculées de manière à vous présenter le plus de résultats possibles. Les quelques graphiques ci-après vous en expliquent les principes.

tab

Le déplacement des fenêtres

`ins´´sel´´pup´´pdo´ sur vt220 ou
`bs´´lf´´pf2´´pf3´ sur rainbow

Tout ce qui sera précisé sur les touches "fenêtres" VT220 restent vrai pour les autres touches (rainbow). Remarquons cependant que les touches du VT220 sont bien mieux placées que celles du rainbow. Je pense que des VT220 (ou compatibles) sont indispensables pour respecter l'ergonomie de ce programme.

Les touches se trouvent juste au-dessus des flèches et dans les mêmes positions relatives. En poussant sur l'une de ces touches, la fenêtre se déplace dans la direction demandée. Si vous vous trouvez sur un des bords du tableau et que vous cherchez à vous déplacer vers l'extérieur de celui-ci, un message d'erreur vous est présenté et la fenêtre ainsi que le curseur ne changent pas de place. Si par contre le déplacement est possible, le curseur se trouve sur l'élément courant si celui-ci existe toujours dans la nouvelle fenêtre, sur le premier résultat rencontré dans le sens du déplacement sinon.

Les déplacements rapides

Ces déplacements rapides se font en poussant deux touches successivement.

Les déplacements rapides dans la fenêtre

`!´´flèches´ : le curseur se déplace sur le résultat extrémal de la fenêtre, dans la direction de la

flèche poussée.

Les déplacements rapides dans le tableau.

`'´`déplacement fenêtre´ : le curseur se déplace sur le résultat extrémal du tableau, dans la direction demandée.

A ce niveau vous pouvez pousser sur la touche `PF1´ -première ligne du pavé numérique- elle vous permet d'accéder aux autres fonctions de ce programme.
Examinons-les en détails

'?' : Cette fonction permet de vous déplacer rapidement sur un résultat du tableau en donnant le numéro de ligne et le numéro de colonne de celui-ci. Si vous ne donnez aucun numéro de ligne (colonne) ou si vous introduisez une suite de caractères ne représentant pas un nombre, la ligne (colonne) courante est prise par défaut. Si vous donnez un nombre en dehors des possibilités du tableau, la position extrémale est prise par défaut. Si vous introduisez des caractères numériques puis d'autres , seul les caractères numériques sont pris en compte. S'il est nécessaire de déplacer la fenêtre, le calcul de celle-ci se fera en essayant de positionner le résultat recherché au centre de l'écran.

'N' : Cette fonction vous permet de vous déplacer rapidement sur le résultat de la première colonne associé à un numéro d'identifiant que vous avez donné Si le numéro d'identifiant n'existe pas ou des caractères autres que des caractères numériques ont été

introduits, un message d'erreur est proposé et aucun déplacement n'est effectué.

'F' : Recherche d'un résultat particulier. Cette recherche se fait sur les résultats qui se trouvent de la position courante à la fin du tableau en sachant qu'un parcours ligne par ligne est effectué. Si le résultat que vous introduisez n'existe pas ou n'est pas conforme au format d'un réel, un message d'erreur vous est proposé et aucun déplacement n'est effectué. Remarque importante. La recherche se fait indifféremment sur un réel, un entier ou un fractionnaire. MAIS si vous recherchez le résultat 1/8 vous devez introduire 8 comme élément recherché.

'R' : vous avez décidé de modifier les résultats de toute une ligne. Vous pouvez supprimer toute une ligne de résultats en pressant la touche '-'. C'est la ligne courante qui est supprimée. Toutes les lignes suivantes sont alors remontées d'une ligne et la dernière ligne est sans résultat. Les résultats ainsi supprimés ne sont pas définitivement perdus, ils se trouvent en fait dans un buffer. Vous pourrez donc récupérer le contenu grâce à une autre fonction qui sera examinée plus tard. Vous pourrez également ajouter une ligne de résultats en poussant sur '+'. Une ligne est ajoutée, précédant la ligne courante. Cette nouvelle ligne est bien sûr sans aucun résultat, les lignes de résultats suivantes sont descendues d'une ligne et la dernière ligne est mémorisée dans le buffer. Si vous pressez toute autre touche, aucune modification n'est effectuée.

'B' : Vous cherchez à récupérer les résultats de la ligne sauvée dans la fonction précédente. Vous substituez les résultats de la ligne courante par les résultats contenus dans le buffer. Les résultats de la ligne courante sont définitivement perdus.

'I' : Vous avez décidé d'introduire l'ensemble des résultats manquants. Cette fonction d'introduction automatique des résultats manquants va vous permettre de compléter les résultats du tableau en vous déplacant de résultat manquant en résultat manquant et en attendant l'introduction de celui-ci. L'introduction des résultats se fait par l'utilisation d'une fonction spécialisée supposée connue par l'utilisateur. Une modification toutefois: si vous tapez un point ('.') en première position, la fonction d'introduction automatique est stoppée et un message d'avertissement vous est affiché. Le déplacement se fait ligne par ligne et ce jusqu'à la fin du tableau ou l'arrêt volontaire de l'utilisateur.

'V' : Vous avez décidé de vérifier et corriger les résultats douteux. Cette fonction vous permet de vous déplacer de résultat douteux en résultat douteux depuis le premier résultat jusqu'au dernier ou l'arrêt volontaire de l'utilisateur. Voici les posssibilités qui vous sont offertes lorsque la fonction se trouve sur un résultat.

`space´ : ceci permet de passer au résultat douteux suivant sans aucune modification de celui-ci.

`PF1´ : arrêt de l'utilisation de cette fonction.

`return´ : force un résultat douteux à devenir normal (un résultat anormal est considéré comme accepté, un

résultat manquant est supprimé -la base de données est modifiée dans ce sens- , un résultat superflu est accepté -la base de données est modifiée dans ce sens-

`back slash´ : ceci permet de supprimer un résultat. (utilisé spécialement pour les résultats superflus)

`autre´ : introduction d'un résultat. Remarque: si le résultat introduit est anormal, il ne sera pas corrigé. Une autre manière de stopper cette fonction est d'introduire un point ('.') en première position.

'Q','E','P' : Il vous reste à quitter le programme. Vous pouvez le faire en enregistrant les modifications que vous venez d'apporter ('E' ou 'P') ou en ne les sauvant pas ('Q'). Lors du sauvetage, il est possible que des problèmes de concurrence entre fichiers apparaissent. Dans ce cas le programme cherche à le résoudre et s'il n'y arrive pas, il demande à l'utilisateur s'il doit essayer à nouveau ou non.

Voilà un tour complet des fonctions actuellement disponibles. Une autre devrait bientôt en faire partie : l'introduction de commentaires liés à un résultat.

Si vous avez quelques critiques quant à l'utilisation de ce programme, je vous prierais de me les faire savoir: j'essaierais d'en tenir compte pour les versions suivantes. Soyez assurés que toutes critiques seront étudiées avec le plus grand intérêt.

Utilisation proposée de ce programme

Je vous propose de n'utiliser ce programme qu'en fin de journée, quand tous les résultats ont été transmis. La première chose à vérifier est la correspondance entre les numéros identifiants à l'écran et ceux transmis par les laborantins. Si cette correspondance n'est pas exacte, consultez les solutions aux problèmes(prochain paragraphe). Si par contre cette correspondance est exacte, vous pourrez utiliser la procédure de vérification automatique. Celle-ci devrait pouvoir vous permettre de corriger rapidement les quelques résultats non-conformes. Il est à remarquer que ces résultats devraient être assez rares. Si ceux-ci devaient être nombreux, redoublez de vigilance; quelque chose d'anormal s'est certainement produit. Quand tous les résultats ont été ainsi vérifiés, sauvez-les dans la base de données par la fonction 'E'. Vous pourrez alors entrer dans un programme d'introduction normal des résultats pour vous permettre d'ajouter les commentaires si nécessaire.

Les quelques problèmes qui peuvent survenir

Concurrence entre fichiers.

Le programme, faisant appel à de nombreux fichiers, peut être confronté à des problèmes de concurrence. Ces problèmes sont résolus de deux manières distinctes :

- Si ce problème se présente à l'initialisation, un message d'erreur vous est présenté et le programme est stoppé aussitôt. Vous devrez donc chercher à supprimer ce problème de concurrence avant de relancer le programme.
- Si cela se passe à la terminaison, une autre politique est proposée. Pour respecter le travail réalisé, le programme cherche à résoudre ce problème de concurrence. S'il n'y parvient pas, il propose à l'utilisateur de réessayer ou de quitter le programme sans modification. Cette possibilité permet de stopper les programmes concurrents avant de réessayer.

Aucun élément ne se trouve dans la JBL.

Le programme d'acceptation ne peut vous servir en rien. Dans ce cas, un message vous est présenté et le programme est stoppé.

Mauvaise programmation.

Ceci est bien sûr exceptionnel ...

- le secrétariat a oublié de programmer une analyse. Le résultat transmis apparaît alors en clignotant (superflu). En utilisant le programme de vérification automatique, vous pourrez valider ce résultat. La demande sera alors complétée automatiquement.
- le secrétariat a programmé une analyse en trop. Le

résultat n'est pas transmis par le ONLINE et apparaîtra sous forme de barres horizontales. L'utilisation de la fonction de vérification automatique vous permettra de supprimer ce résultat. La demande sera alors automatiquement imputée de cette analyse.

- le labo a oublié de programmer une analyse. Ce résultat apparaîtra sous forme de barres horizontales. Il faudra alors demander cette analyse en urgence et l'introduire manuellement soit via ce même programme dans une utilisation ultérieure, soit via un autre programme d'introduction de résultats.

- le labo a programmé une analyse en trop. Le résultat apparaîtra en clignotant. Vous devez alors le supprimer. Vous avez alors plusieurs possibilités : utiliser la fonction de vérification et le supprimer ou alors utiliser la fonction d'introduction et supprimer tous les caractères du champs.

- le secrétariat a oublié de programmer toute une demande. Je vous propose la manière de suivre suivante:

. vous sortez du programme par 'Q'

. vous introduisez la demande en dernière position et vous refaites les JBL

. vous retournez dans le programme d'acceptation

. vous vous placez sur la ligne de résultats qui ne correspond plus à son identifiant

. vous supprimez cette ligne de résultats. Dans ce cas, les autres lignes de résultats devraient se retrouver en face de leur identifiant.

. vous vous déplacez sur le dernier identifiant (celui que vous venez d'introduire)

. Vous recopiez le buffer.

Vous devriez, dès lors, avoir attribué tous les

résultats à leur identifiant. Remarque: dans ce cas, les numéros de godet ne correspondent plus.

- le labo a oublié de programmer toute une demande. Je vous propose la manière de suivre suivante.

. vous quittez le programme par 'Q'

. vous demandez au labo de faire les analyses en utilisant la position suivant la dernière utilisée.

. vous retournez dans le programme après transfert des résultats

. vous vous placez sur la ligne de résultats ne correspondant plus à son identifiant.

. vous insérez une ligne de résultats. Dans ce cas la dernière ligne du ONLINE a été mémorisée dans le buffer.

. vous recopiez le buffer

Vous devriez avoir attribué les résultats à leur identifiants

Format complètement incohérent.

Ceci peut être dû à des problèmes sur l'UNINA, des problèmes lors du transfert (faux contacts dans les prises) ou encore à des perturbations électromagnétiques (orage, moteur électrique ...). Dans ce cas, je vous suggère d'introduire les résultats manuellement en utilisant un autre programme.

Vous cherchez à introduire un résultat incompatible avec le format autorisé (un résultat plus grand que permis, ou d'un autre type).

Dans ce cas, vous devez modifier le format dans SIGANA avant de pouvoir continuer.

Les résultats de toute une colonne ne sont pas transmis ou ne correspondent pas à la colonne dans la JBL.

Vérifiez par le programme 'modif_param' la correspondance numéro d'analyse / numéro de colonne. Si tout semble correct, demandez au laborantin s'il n'a pas changé les analyses. De toute façon, ces résultats sont perdus.

Aucun résultat n'est transmis.

Vérifier que le système UNINA est toujours en état de marche et que les résultats sont bien transmis. Vous serez alors obligé d'introduire tous les résultats à la main.

Plantage pur et simple.

Je vous demande alors de mettre sur papier les dernières opérations effectuées et de recopier le message d'erreur dans son entièreté. Si je me trouve dans la maison, faites appel à moi. Je vous propose de réessayer le programme et si la même erreur est commise vous devrez compléter les résultats par un autre système.

P.Wautié Mars 1987

RESUME COMMANDES VALIDATION

<Flèches> : déplacement curseur.

<INS> <SEL> <PUP> <PDO>
: déplacement fenêtre
<BS> <LF> <PF2> <PF3>

<'><Flèches>
: déplacement rapide
<'> <BS> <LF> <PF2> <PF3>

<PF1> <?> : positionnement sur un élément du tableau

<PF1> <N> : positionnement sur un numéro de réference

<PF1> <F> : recherche d'un résultat

<PF1> <R> <-> : suppression d'une ligne de résultats

<PF1> <R> <+> : ajout d'une ligne de résultats

<PF1> <B> : remplacement de la ligne courante par le buffer

<PF1> <I> : introduction automatique

<PF1> <V> : vérification automatique

<PF1> <Q> : sortie sans sauvetage des modifications

<PF1> <E> : sortie avec transfert dans la base de données

<PF1> <P> : sortie avec sauvetage des modifications mais sans transfert dans la base de données

MODE D'EMPLOI DU PROGRAMME DE CONNECTION

VAX-LAB

MODE D'EMPLOI DU LOGICIEL DE CONNECTION ON_LINE

P.Wautié.

MARS 1987

MODE D'EMPLOI DE 'CONNECTION'

But du programme

Ce programme permet la réception et l'acceptation de données transmises par le concentrateur UNINA.

Etude des différentes phases du programme

Aucun mode d'emploi n'est vraiment nécessaire; dès que ce programme est lancé, il se suffit à lui-même et ne demande plus aucune intéraction.

Les quelques problèmes qui peuvent survenir

Le système UNINA semble être bloqué.

Vérifiez

- que la vitesse du concentrateur est bien de 9600 bps.
- que le programme se déroule toujours.
- que les prises ne sont pas déconnectées.

Les résultats sont transmis mais ne sont pas disponibles par le programme de validation.

Vérifiez que le fichier de paramètres 'param_online' est toujours accessible dans la directory 'on_line'.

P.Wautié Mars 1987

MODE D'EMPLOI DU PROGRAMME DE MODIFICATION DE PARAMETRES

VAX-LAB

MODE D'EMPLOI DU LOGICIEL DE MODIFICATION DES PARAMETRES DES MACHINES D'ANALYSES ON_LINE

P.Wautié.

MARS 1987

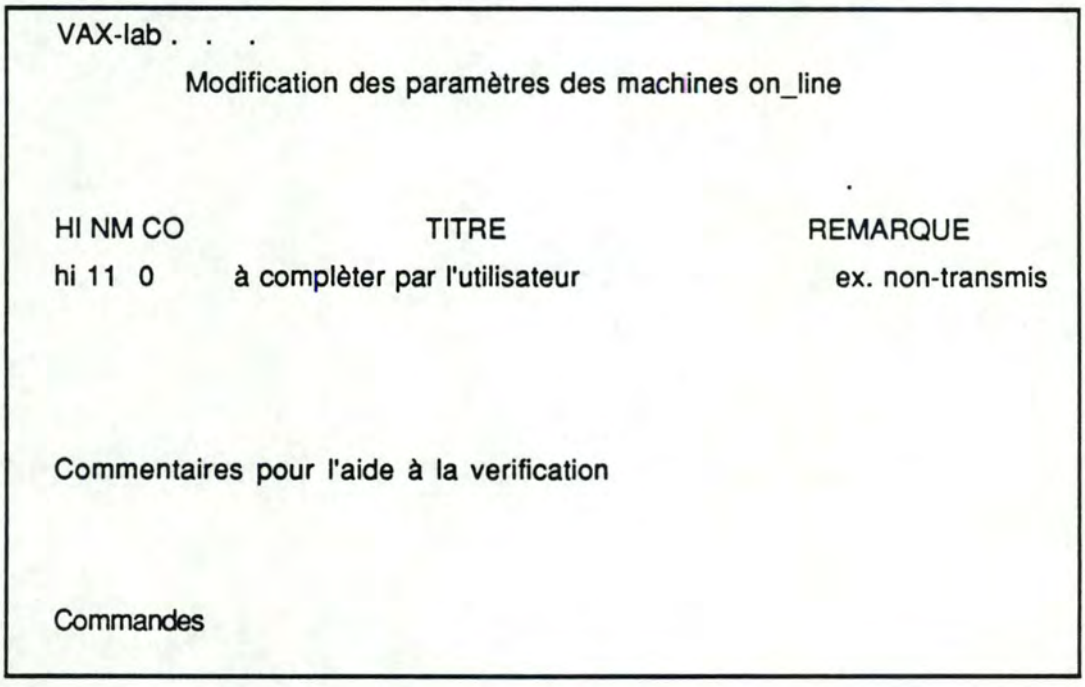
MODE D'EMPLOI DE 'MODIF_PARAM'

But du programme

Le programme de connection du ON-LINE necessite beaucoup de paramètres. Ce programme permet de modifier facilement quelques-uns de ceux-ci.

Etudes des différentes phases du programme

Le programme débute après l'instruction "run modif_parm". Je vous présente ci-après le premier écran en vous précisant toutes les caractéristiques de celui-ci.



a) Titre général

Titre commun à toutes les fonctions du programme 'LABO'.

b) Titre du programme

Celui-ci est "Modification des paramètres des machines ON-LINE".

c) La ligne suivante est une ligne de présentation dont voici la signification:

'id' : identification en deux lettres de la machine d'analyses concernée (HI = Hitachi , CL = Coulter , BN = BNA)

'nm' : ceci a deux significations différentes: soit la valeur de la ligne courante, soit le numéro de l'analyse transmis par la machine.

'co' : numéro de colonne vers où le résultat de l'analyse 'nm' doit aparaître.

'titre' : commentaires libres que vous pouvez ajouter.

'rem' : remarques qui sont données par le programmeur ou le programme.

d) Ligne courante de paramètres

e) Quelques lignes de commentaires qui apparaissent à chaque modification

f) Ligne de commandes utilisables

Utilisation

Ce programme vous permet de modifier

- la correspondance entre le numéro de l'analyse et le numéro de colonne dans laquelle le résultat doit apparaître lors de l'acceptation. (Le numéro d'analyse est fixé par la machine d'analyses).
- la correspondance entre la machine d'analyses et le numéro de JBL.
- un commentaire qui peut vous aider lors de l'utilisation ultérieure de ce programme.

Vous ne pouvez modifier que les valeurs de la ligne de paramètres qui vous sont proposés. C'est en fonction de celle-ci que vous pourrez modifier soit le numéro de colonne de l'analyse, soit le numéro de JBL (une remarque vous fait savoir que vous vous trouvez sur un numéro de JBL). Pour passer de ligne en ligne, vous devez taper '2' ou '8'. Vous pourrez modifier le commentaire de la ligne courante en tapant sur la touche 'c', ou encore modifier le paramètre de correspondance en tapant la lettre 'n'.

La procédure d'introduction de données est supposée connue. Après modification d'un des paramètres, quelques lignes vous sont proposées pour vous aider à la vérification. Elles vous donnent le résultat des modifications que vous venez d'apporter.

Pour sortir du programme, vous avez deux possibilités: en acceptant les modifications que vous venez d'effectuer ('E') ou en les oubliant ('Q').

Les quelques problèmes qui peuvent survenir

Le programme vérifie que tous les fichiers sont correctement épurés.

Si ce n'est pas le cas, un message d'erreur vous est présenté et le programme stoppé.

Des problèmes de concurrences entre fichiers peuvent survenir.

Avant de relancer ce programme vérifiez que les autres programmes sont terminés et le programme de connection arrêté.

Les modifications ne semblent pas être prise en compte.

Vérifiez alors que le programme de connection était bien stoppé lors des modifications et que les modifications étaient sauvées.

Plantage pur et simple.

Je vous demande alors de mettre sur papier les dernières opérations effectuées ainsi que le message d'erreur proposé. Faites alors appel à un des informaticiens de la maison.

Améliorations

Ce programme peut être modifié et amélioré de multiples façons (trop nombreuses à énumérer). Mais, l'utilisation de ce programme reste exceptionnelle, de plus, les paramètres sont sous format d'un fichier texte ce qui permet de l'éditer par un éditeur classique. C'est pourquoi je pense qu'il n'est peut-être pas nécessaire d'apporter ces améliorations

P.Wautié Mars 1987

MODE D'EMPLOI DU PROGRAMME D'INTRODUCTION DE RESULTATS

VAX-LAB

MODE D'EMPLOI DU LOGICIEL D'INTRODUCTION DES DONNEES PAR JBL

P.Wautié.

MARS 1987

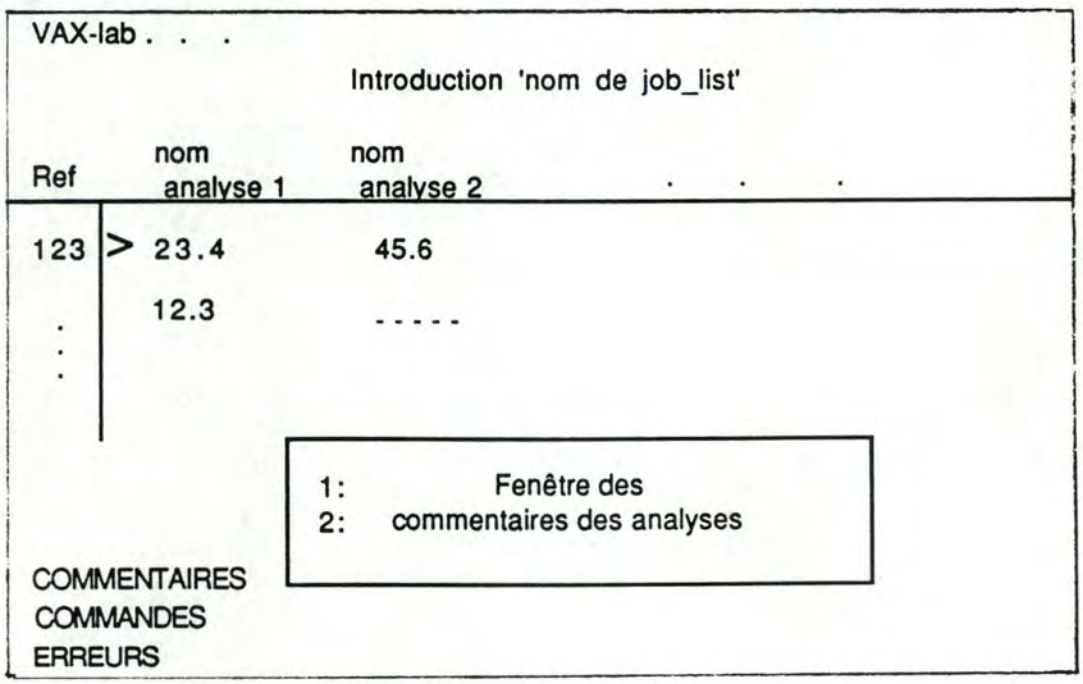
MODE D'EMPLOI DE 'INTRODUCTION'

But du programme

Ce programme permet une introduction des résultats par job-liste.

Etudes des différentes phases du programme

Le programme est lancé via un système de choix multiples. Un premier écran vous est proposé ainsi qu'un message vous demandant de patienter quelques secondes. Il est à remarquer que l'attente peut durer assez longtemps. Patience donc ... L'initialisation terminée, l'écran se complète et affiche une partie des résultats de la job-liste. Je vous présente ci-après le premier écran en vous précisant toutes les caractéristiques de celui-ci.



a) Titre général

Titre commun à toutes les fonctions du programme 'LABO'.

b) Titre du programme

Ce titre est le nom de la fonction dans ce programme 'LABO'. Il est composé de deux choses:

- du terme introduction indiquant qu'on se trouve bien dans le programme d'introduction par job-liste.
- du titre de la JBL concernée par cette introduction.

c) La ligne suivante est composée de signes dont voici la signification

'ref' : donne la colonne des numéros identifiants des demandes d'analyses. (000000 - 999999)

les noms d'analyses : cinq lettres maximum donnent le code de l'analyse. Ces codes sont définis par ailleurs (sigana.dat).

d) Le tableau

Les résultats déjà introduits sont dans la matrice ayant pour abscisses les noms des analyses et pour ordonnées les numéros d'identifiants. Les résultats sont de trois types différents

- anormaux : les résultats sont en-dehors des normes de vraissemblance consacrées à cette analyse pour des sujets sains. (Ils apparaissent en gras à l'écran)
- manquants : ce sont les résultats programmés par le secrétariat mais qui n'ont pas encore été introduit (ceux-ci sont représentés par des barres horizontales)
- non-douteux : ce sont les résultats qui semblent être corrects (ni anormaux, ni manquants)

e) Le curseur

Ce curseur est représenté par un '´' clignotant. Ce curseur donne la position courante, c'est-à-dire donne le résultat, la ligne ou la colonne sur lesquels des modifications peuvent être effectuées.

f) La ligne de commentaires

Cette ligne reprend les commentaires standards du résultat courant.

g) La ligne de commandes

Cette ligne reprend le nom de la fonction dans laquelle on se trouve ainsi que toutes les possibilités liées à celle-ci. Si les possibilités ne tiennent pas en une seule ligne, il est possible de voir les autres en poussant sur `tab´. Si le fichier de commandes est inaccessible, la ligne est remplacée par 'no comment'

h) Ligne d'erreur et d'avertissement

Celle-ci est la dernière de l'écran et reprend toutes les erreurs et possibilités d'avertissement. Si le fichier des erreurs n'est pas accessible, la ligne 'undefined error' est présentée en cas d'erreur.

Première possibilité : le déplacement

Voilà vous vous trouvez dans un programme en état de marche (aucun problème n'est apparu lors de l'initialisation). La première possibilité qui vous est offerte est le déplacement du curseur ou des fenêtres. Vous pouvez voir sur la ligne de commandes, l'ensemble des possibilités qui existent dans ce niveau. Examinons-les en détail.

Le déplacement curseur.

`les flèches´ : celles-ci se trouvent sur la droite du clavier alphabétique et sur la gauche du bloc numérique. En poussant sur l'une de ces flèches, le curseur se déplace d'un résultat dans la direction de la flèche. Si vous vous trouvez sur un bord du tableau, et si vous cherchez à vous déplacer vers l'extérieur de celui-ci, un message d'erreur vous est présenté et le curseur ne quitte pas sa position d'origine. Si vous vous trouvez sur le bord d'une fenêtre sans être sur le bord du tableau, un déplacement curseur donne lieu à la présentation d'une nouvelle fenêtre et à un positionnement sur l'élément demandé (flèches). Les fenêtres sont calculées de manière à vous présenter le plus de résultats possibles. Les quelques graphiques ci-après vous en expliquent les principes.

tab

Le déplacement des fenêtres

`ins´´sel´´pup´´pdo´ sur vt220 ou
`bs´´lf´´pf2´´pf3´ sur rainbow

Tout ce qui sera précisé sur les touches "fenêtres" VT220 restent vrai pour les autres touches (rainbow). Remarquons cependant que les touches du VT220 sont bien mieux placées que celles du rainbow. Je pense que des VT220 (ou compatibles) sont indispensables pour respecter l'ergonomie de ce programme.

Les touches se trouvent juste au-dessus des flèches et dans les mêmes positions relatives. En poussant sur l'une de ces touches, la fenêtre se déplace dans la direction demandée. Si vous vous trouvez sur un des bords du tableau et que vous cherchez à vous déplacer vers l'extérieur de celui-ci, un message d'erreur vous est présenté et la fenêtre ainsi que le curseur ne changent pas de place. Si par contre le déplacement est possible, le curseur se trouve sur l'élément courant si celui-ci existe toujours dans la nouvelle fenêtre, sur le premier résultat rencontré dans le sens du déplacement sinon.

Les déplacements rapides

Ces déplacements rapides se font en poussant deux touches successivement.

Les déplacements rapides dans la fenêtre

`+´´flèches´ : le curseur se déplace sur le résultat extrémal de la fenêtre, dans la direction de la

flèche poussée.

Les déplacements rapides dans le tableau.

`''déplacement fenêtre´ : le curseur se déplace sur le résultat extrémal du tableau, dans la direction demandée.

L'introduction d'un résultat est supposée connue. Une modification lui a été apportée : vous pouvez introduire des commentaires standards en les faisant précéder par le caractère '&'. D'autre part vous pouvez accéder à la fenêtre de commentaire en appuyant sur la flèche vers le bas.

A ce niveau vous pouvez pousser sur la touche `PF1´ première ligne du pavé numérique, elle vous permet d'accéder aux autres fonctions de ce programme.
Examinons-les en détails

'?' : Cette fonction permet de vous déplacer rapidement sur un résultat du tableau en donnant le numéro de ligne et le numéro de colonne de celui-ci. Si vous ne donnez aucun numéro de ligne (colonne) ou si vous introduisez une suite de caractères ne représentant pas un nombre, la ligne (colonne) courante est prise par défaut. Si vous donnez un nombre en dehors des possibilités du tableau, la position extrémale est prise par défaut. Si vous introduisez des caractères numériques puis d'autres , seul les caractères numériques sont pris en compte. S'il est nécessaire de déplacer la fenêtre, le calcul de celle-ci se fera en essayant de positionner le résultat

recherché au centre de l'écran.

'N' : Cette fonction vous permet de vous déplacer rapidement sur le résultat de la première colonne associé à un numéro d'identifiant que vous avez donné Si le numéro d'identifiant n'existe pas ou des caractères autres que des caractères numériques ont été introduits, un message d'erreur est proposé et aucun déplacement n'est effectué.

'F' : Recherche d'un résultat particulier. Cette recherche se fait sur les résultats qui se trouvent de la position courante à la fin du tableau en sachant qu'un parcours ligne par ligne est effectué. Si le résultat que vous introduisez n'existe pas ou n'est pas conforme au format d'un réel, un message d'erreur vous est proposé et aucun déplacement n'est effectué. Remarque importante. La recherche se fait indifféremment sur un réel, un entier ou un fractionnaire. MAIS si vous recherchez le résultat 1/8 vous devez introduire 8 comme élément recherché.

'R' : vous avez décidé de modifier les résultats de toute une ligne. Vous pouvez supprimer toute une ligne de résultats en pressant la touche '-'. C'est la ligne courante qui est supprimée. Toutes les lignes suivantes sont alors remontées d'une ligne et la dernière ligne est sans résultat. Les résultats ainsi supprimés ne sont pas définitivement perdus, ils se trouvent en fait dans un buffer. Vous pourrez donc récupérer le contenu grâce à une autre fonction qui sera examinée plus tard. Vous pourrez également ajouter une ligne de résultats en poussant sur '+'. Une ligne est ajoutée,

précédant la ligne courante. Cette nouvelle ligne est bien sûr sans aucun résultat, les lignes de résultats suivantes sont descendues d'une ligne et la dernière ligne est mémorisée dans le buffer. Si vous pressez toute autre touche, aucune modification n'est effectuée.

'B' : Vous cherchez à récupérer les résultats de la ligne sauvée dans la fonction précédente. Vous substituez les résultats de la ligne courante par les résultats contenus dans le buffer. Les résultats de la ligne courante sont définitivement perdus.

'I' : Vous avez décidé d'introduire l'ensemble des résultats manquants. Cette fonction d'introduction automatique des résultats manquants va vous permettre de compléter les résultats du tableau en vous déplacant de résultat manquant en résultat manquant et en attendant l'introduction de celui-ci. L'introduction des résultats se fait par l'utilisation d'une fonction spécialisée supposée connue par l'utilisateur. Une modification toutefois: si vous tapez un point ('.') en première position, la fonction d'introduction automatique est stoppée et un message d'avertissement vous est affiché. Le déplacement se fait ligne par ligne et ce jusqu'à la fin du tableau ou l'arrêt volontaire de l'utilisateur.

'Q','E' : Il vous reste à quitter le programme. Vous pouvez le faire en enregistrant les modifications que vous venez d'apporter ('E') ou en ne les sauvant pas ('Q'). Lors du sauvetage, il est possible que des problèmes de concurrence entre fichiers apparaissent.

Dans ce cas le programme cherche à le résoudre et s'il n'y arrive pas, il demande à l'utilisateur s'il doit essayer à nouveau ou non.

Si vous avez quelques critiques quant à l'utilisation de ce programme, je vous prierais de me les faire savoir, j'essaierais d'en tenir compte pour les versions suivantes. Soyez assurés que toutes critiques seront étudiées avec le plus grand intérêt.

<Flèches> : déplacement curseur.

<INS> <SEL> <PUP> <PDO>
: déplacement fenêtre
<BS> <LF> <PF2> <PF3>

<'><Flèches>
: déplacement rapide
<'> <BS> <LF> <PF2> <PF3>

<ESC> : fenêtre de commentaires

<PF1> <?> : positionnement sur un élément du tableau

<PF1> <N> : positionnement sur un numéro de réference

<PF1> <F> : recherche d'un résultat

<PF1> <R> <-> : suppression d'une ligne de résultats

<PF1> <R> <+> : ajout d'une ligne de résultats

<PF1> <B> : remplacement de la ligne courante par le buffer

<PF1> <I> : introduction automatique

<PF1> <Q> : sortie sans sauvetage des modifications

<PF1> <E> : sortie avec transfert dans la base de données

ERREUR

======

Suite de lignes d´erreur des programmes de validation et d´introduction par JBL.

ERREUR : il n'existe pas de ligne(s) precedente(s).
ERREUR : il n'existe pas de ligne(s) suivante(s).
ERREUR : il n'existe pas de colonne(s) precedente(s).
ERREUR : il n'existe pas de colonne(s) suivante(s).
ERREUR : valeur non trouvee.
ERREUR : numero d'identification inexistant dans ce tableau.
ERREUR : tableau vide ou JBL inexistante : stop. <return>
ERREUR LECTURE FICHIER : <Q> : quitter, <autre> : reessayer
AVERTISSEMENT : introduction automatique terminee
AVERTISSEMENT : verification automatique terminee
ERREUR : format incorrect
ERREUR : code de remarque standard inconnu
ERREUR : commentaire "court" apres des commentaires "long"
ERREUR : pas de resultat (ou commentaire) pour cette analyse
ERREUR : transfert en cours ! recommencez dans quelques minutes

COMMENTAIRES

Les commentaires liés aux fonctions des programmes de validation et d´introduction par JBL. Le nombre de lignes de commentaire est quelconque. Il est donc necessaire de terminer un commentaire par un "END".

```
DEPLACEMENT : <fleches> : directions, <TAB>,
DEPLACEMENT : <'> <fleches> : bornes ecran, <TAB>
DEPLACEMENT : <bs>, <lf>, <PF2>, <PF3> : deplacement fenetre, <TAB>
DEPLACEMENT : OU <ins>, <sel>, <pup>, <pdo> : deplacement fenetre, <TAB>
DEPLACEMENT : <'> + <deplacement fenetre> : bornes du tableau, <TAB>
DEPLACEMENT : <PF1> : fonction, <TAB>
end
INTRO_AUTO : introduction d'un resultat  <RET> <TAB>    FORMAT :
INTRO_AUTO : un point en PREMIERE position : stop <TAB> FORMAT :
end
CORRECT_AUTO : <space> : aucune modification ; <\> : suppression,<TAB>
CORRECT_AUTO : <return> : acceptation ; <PF1> : stop , <TAB>
CORRECT_AUTO : <autre> : introduction, <TAB>
end
LIGNE DE RESULTAT : tapez <-> pour supprimer, <+> pour ajouter,<TAB>
LIGNE DE RESULTAT : <autre> : annulation de la fonction, <TAB>
end
RECHERCHE RESULTAT : donnez le resultat a rechercher :
end
RECHERCHE NUMERO : donnez l'identifiant a rechercher :
end
SAUT : donnez le numero de ligne (ligne courrante par defaut) :
SAUT : donnez le numero de colonne (colonne courante par defaut ) :
end
INITIALISATION : veuillez patienter quelques instants svp
end
INTRODUCTION D'UN RESULTAT  <RET>                FORMAT :
end
FONCTION : E(xit), Q(uit), F(ind res), I(ntroduction auto), <TAB>
FONCTION : V(erification auto), ligne de R(es), <TAB>
FONCTION : ?(saut), recopie du B(uffer), C(ommentaire), <TAB>
FONCTION : N(umero ident)  <autre> : retour a deplacement, <TAB>
FONCTION : sauvetage P(artiel)
end
EXIT : sauvetage en cours ! quelques minutes de patience ...
end
INITIALISATION : choix d'une JBL: <up> , <down> , <return>
end
```

PARAM-ONLINE

==========

Liste des paramètres communs aux programmes de validation et de connection.

```
hi 9629 39**.** num JBL
cl 10198 42**.** num JBL
bn 42 43**.** num JBL
hi  1  0     . ACP
hi  2  0     . ALB
hi  3  0 NON_TRANSMIS    . ALC
hi  4  0     . ALDOL
hi  5  8 PHOSPHATASE ALCALINE     . ALP
hi  6  0     . AMMN
hi  7 14 AMYLASE     . AMYL
hi  8  1 UREE     . BUN
hi  9  0     . CA
hi 10 16 CHOLINESTERASE     . CHE
hi 11  4 CHOLESTEROL     . CHOL
hi 12  0     . CK_MB
hi 13 13     . CPK
hi 14  0     . CREA
hi 15  0     . D_BIL
hi 16  7 GGT     . GGT
hi 17  0     . GLDH
hi 18  0     . GLU
hi 19 10     . GOT
hi 20 11     . GPT
hi 21  0     . HBDH
hi 22  0     . HDL
hi 23  0     . IRON
hi 24  0     . LA
hi 25  9     . LAP
hi 26 12     . LDH
hi 27  0     . LIPAS
hi 28  0     . MG
hi 29  5 PHOSPHOLIPIDES     . PHOS
hi 30  0     . SAL
hi 31  0     . T_BIL
hi 32  6 PROTEINE TOTALE     . TP
hi 33  3 TRIGLYCERIDES     . TRIG
hi 34  2 ACIDE URIQUE     . UA
hi 35  0     . UIBC
hi 36 15 PHOSPHORE     . A
hi 37  0     . CO2
hi 38 17 SODIUM     . NA
hi 39 18 POTASSIUM     . K
hi 40 19 CHLORE     . CL
cl  1  6 GB     . WBC
cl  2  2 GR     . RBC
cl  3  1 HB     . HGB
cl  4  3 HT     . HCT
cl  5  4 VG     . MCV
cl  6  8 MEN_TRANSMIS    . MCH
cl  7  5 CCMH     . MCHC
bn  1 14     .
bn  2 15     .
bn  3 16     .
bn  4 17     .
bn  5 18     .
bn  6  1     .
bn  7  4     .
bn  8  5     .
bn  9  6     .
bn 10  7     .
bn 11  8     .
bn 12  0     .
bn 13  0     .
bn 14  0     .
bn 15  9     .
bn 16  0     .
```

```
bn 18  0     .
bn 19  0     .
bn 20  0     .
bn 21 10     .
bn 22  0     .
bn 23  0     .
bn 24  0     .
bn 25  0     .
bn 26  0     .
bn 27  0     .
bn 28  0     .
bn 29  0     .
bn 30  0     .
bn 31  0     .
bn 32 11     .
bn 33 12     .
bn 34  0     .
bn 35  2   .
bn 36  3   .
bn 37  19    .
bn 38  0     .
bn 39  0     .
bn 40  0     .
bn 41  0     .
bn 42  0     .
bn 43  0     .
bn 44  0     .
bn 45 13     .
bn 46  0     .
bn 47  0     .
bn 48  0     .
bn 49  0     .
bn 50  0     .
bn 51  0 NON_TRANSMIS    .
bn 52  0     .
bn 53  0     .
bn 54  0     .
bn 55  0     .
bn 56  0     .
bn 57  0     .
bn 58  0     .
bn 59  0     .
bn 60  0     .
bn 61  0     .
bn 62  0     .
bn 63  0     .
bn 64  0     .
bn 65  0     .
bn 66  0     .
bn 67  0     .
bn 68  0     .
bn 69  0     .
bn 70  0     .
```

SET-TERM

========

Suite d´instructions DCL necessaire à l´initialisation de la porte de communication.

```
allocate txa6: online /nolog
set term txa6: /speed=9600 /noecho /passall /pasthru /par=even /noeightbit
```

JOB-LISTE ORDINATEUR (JBL ORDI)

Liste de travail obtenue après encodage des demandes.

			HDW	PTT	CEP	FIB	TS	TC	PDF	TT	COMMENTAIRE
011710 Guery	F	33				...					
011712 Wante	F					...					
011713 Compere	H	43				...					
011718 Durant	F	28		...		...					
011719 Pierart	H	50				...					
011721 Lachat	H	29		...		...					
011723 Lambillon	F	53				...					
011733 Rouchy	H	58				...					
011735 Paglia	H	26		...		...					
011736 Di Stante	F	21				...					
011738 Henin	F	41	...	...	...	...				...	
011739 Sampoux	F	58				...					
011744 Lumanne	H	35				...					
011746 Quairiaux	H	13				...					
011751 Florent	F	61				...					
011752 Fichefet	H	43				...					
011756 Loriaux	H	61		...		...					
011762 Druet	F	90				...					
011765 D'Haeseleer	H	61				...					

011770 F 71 Delroux		...	...							
011775 F 69 Pernet		...		...						
011781 H 27 Reul		...		...						
011784 F 61 Patte		...		...						
011796 F 74 Withof				...						
011803 F 13 Jurion		...	...	...						
011804 H 60 Roisin		...		...						
011808 H 46 Piret				...						
011809 F 49 Hoyoux				...						
011814 F 57 Bertinchamps		...		...						
011818 H 29 Sauvage				...						
011819 H 5 Kraayveld	...	...	...	...					...	
011821 F Berest				...						
011823 F Hanne				...						
011825 H 6 Kraayveld	...	...	...	...					...	
011828 H 65 Istasse				...						
011829 H 61 Thieffry				...						
011831 H 34 Plasman		...								

JOB-LISTE LABO (JBL LABO)

Liste de travail manuel complètée par une personne du laboratoire.

1

Réf.	HOW	PTT	%	CEP	FIBRI	g/l	TS	TC'	PDF	TT			
1924		< 5											
1926				15.5		102							
1928		—					—		—				

DEMANDE D´ANALYSES

==========

Demande type proposée par le laboratoire.

DEMANDE D'ANALYSE DE .....

Date de naissance: ..... Sexe : M - F Hospitalisé: OUI - NON

Renseignements administratifs ou vignette

Assuré(e): .....

Adresse: .....

Mutuelle: .....

N° d'immatriculation: .....

Téléphone: .....

- ☐ A.O.
- ☐ VIPO 75 %
- ☐ VIPO 100 %
- ☐ T.I.: tous risques
- ☐ T.I.: gros risques

HEMATOLOGIE (5 ML)

- ☐ Hémoglobine
- ☐ Globules rouges + Ht
- ☐ Globules blancs
- ☐ Formule
- ☐ Plaquettes
- ☐ Fer
- ☐ Ferritine
- ☐ Transferrine
- ☐ TIBC
- ☐ vit B12
- ☐ Acide folique
- ☐ Réticulocytes
- ☐ Eosinophilie
- ☐ LE cells
- ☐ Electrophorèse de l'hémoglobine

IMMUNOHÉMATOLOGIE

- ☐ Groupe ABO
- ☐ Groupe Rh (D^{u} si Rh −)
- ☐ Sous-groupes Rh
- ☐ Coombs direct
- ☐ Anticorps irréguliers
- ☐ Agglutinines froides

HEMOSTASE

- ☐ Temps de Howell
- ☐ Temps de Quick (PTT)
- ☐ Temps de céphaline
- ☐ Temps de thrombine
- ☐ Antithrombine III
- ☐ Fibrinogène
- ☐ P.D.F.
- ☐ Temps de saignement

CHIMIE (5 ML)

- ☐ Urée
- ☐ Créatinine
- ☐ Acide urique
- ☐ Bilirubine (T + C)
- ☐ Protéines totales
- ☐ Electrophorèse des protéines
- ☐ Glucose
- ☐ Hémoglobine glycosylée
- ☐ Insuline ☐ C-peptide
- ☐ Thymol

IONS (5 ML)

- ☐ Sodium
- ☐ Potassium
- ☐ Chlore
- ☐ Calcium
- ☐ Phosphore
- ☐ Magnésium
- ☐ Magnésium érythrocytaire
- ☐ Cuivre
- ☐ Réserve Alcaline
- ☐ Lithium

BILAN LIPIDIQUE (5 ML)

- ☐ Lipidogramme
- ☐ Cholestérol total
- ☐ Cholestérol HDL
- ☐ Triglycérides
- ☐ Phospholipides
- ☐ Apolipoprotéines A1
- ☐ Apolipoprotéines B

ÉPREUVES FONCTIONNELLES (sur rendez-vous)

- ☐ Hyperglycémie (5T)
- ☐ Courbe d'insulinémie
- ☐ Test au TRH
- ☐ Clearance urée
- ☐ Clearance créatinine
- ☐ Clearance acide urique

PROFIL PROTÉIQUE (5 ML)

- ☐ Albumine
- ☐ α 1 antitrypsine
- ☐ Orosomucoïde
- ☐ Haptoglobine
- ☐ Transferrine
- ☐ C'3
- ☐ IgG
- ☐ IgA
- ☐ IgM
- ☐ CRP
- ☐ Immunoélectrophorèse
- ☐ Céruloplasmine
- ☐ C'4
- ☐ α 2 Macroglobuline
- ☐ β 2 Microglobuline
- ☐ IgE

ENZYMES (5 ML)

- ☐ Gamma GT
- ☐ Phosphatases alcalines
- ☐ 5' Nucléotidase
- ☐ LAP
- ☐ GOT
- ☐ GPT
- ☐ LDH
- ☐ Iso LDH
- ☐ Cholinestérase
- ☐ CPK
- ☐ CPK (MB)
- ☐ Phosphatases acides totales
- ☐ Phosphatase acide prostat.
- ☐ Amylases (sang)
- ☐ Amylases (urine)
- ☐ Lipases

TESTS INFLAMMATOIRES ET RHUMATISMAUX

- ☐ Vitesse de sédimentation
- ☐ Fibrinogène
- ☐ CRP (quantitatif)
- ☐ Orosomucoïde
- ☐ α 1 Antitrypsine
- ☐ Haptoglobine
- ☐ RA test ☐ Waaler-Rose
- ☐ ASL ☐ ASK
- ☐ Facteur Antinucléaire

AUTO-IMMUNITÉ

- ☐ Anticorps antinucléaires
- ☐ Anticorps anti DNA
- ☐ Anticorps anti muscles lisses
- ☐ Anticorps anti mitochondries

HÉPATITE

DIAGNOSTIC

- ☐ Hépatite A - IgM
- ☐ Hépatite B: A.G. HBs
- ☐ A.C. HBc
- ☐ Cytomégalovirus - IgM
- ☐ EBV (IgG - IgM)

STATUT IMMUNITAIRE (Hépatite B)

- ☐ HBs ☐ Anti HBc ☐ Anti HBs
- ☐ HBe ☐ Anti HBe (si HBs+)

IMMUNOLOGIE M.N.I.

- ☐ Paul Bunnell Davidsohn
- ☐ Epstein-Barr (VCA) IgM et IgG
- ☐ M.N.I. test

IMMUNOLOGIE TOXOPLASMOSE

- ☐ IgG par Immunofluorescence
- ☐ IgM par Immunofluorescence
- ☐ par agglutination

IMMUNOLOGIE Σ

- ☐ FTA ☐ VDRL ☐ TPHA

IMMUNOLOGIE BACTÉRIENNE

- ☐ Brucellose (Wright)
- ☐ Salmonellose (Widal)
- ☐ Listériose

SÉROLOGIE ANTI-VIRUS, MYCOPLASMES, CHLAMYDIA

- ☐ Chlamydia Trachomatis
- ☐ Mycoplasma pneumoniae
- ☐ Rubéole (IHA + IgM)
- ☐ Rougeole (IgG - IgM)
- ☐ Varicelle - Zona (IgG - IgM)
- ☐ Oreillons (IgG - IgM)
- ☐ Herpès (HSV 1 - 2) (IgG - IgM)
- ☐ Epstein - Barr (IgG - IgM)
- ☐ Cytomégolavirus (IgM)

SYNDROMES RESPIRATOIRES

- ☐ Mycoplasma pneumoniae
- ☐ Influenza A
- ☐ Influenza B
- ☐ Parainfluenza 1
- ☐ Parainfluenza 2
- ☐ Parainfluenza 3
- ☐ Respiratory syncytial factor
- ☐ Ornithose-Psittacose
- ☐ Adénovirus

SYNDROME CUTANÉ

ADÉNOPATHIES

- ☐ Rubéole
- ☐ Rougeole
- ☐ Varicelle - Zona
- ☐ Oreillons
- ☐ Herpès
- ☐ Cytomégalovirus
- ☐ Epstein - Barr

IMMUNO - ALLERGOLOGIE (Max. 8 allergènes)

MIXTURES

- ☐ gx3 Graminées Mix (g1, g5,g6,g12,g13)
- ☐ wx5 Herbacées Mix I (w1, w6, w7, w8, w12)
- ☐ wx6 Herbacées Mix II (w9, w10, w11, w18)
- ☐ tx5 Arbres Mix I (t2, t4, t8, t12, t14)
- ☐ tx6 Arbres Mix II (t1, t3, t5, t7, t10)
- ☐ mx1 Moisissures Mix (m1, m2, m3, m6)
- ☐ ex1 Epithélia Mix (e1, e3, e4, e5)

ACARIENS

- ☐ d1 ☐ d2

- ☐ **IgE totales**

POUSSIÈRES DE MAISON

- ☐ h1 nº 1 Greer
- ☐ h2 nº 2 Hollister-Stier
- ☐ h3 nº 3 Bencard

ANIMAUX

- ☐ e1 Epithélium de chat
- ☐ e3 Poils et squames de cheval
- ☐ e4 Poils et squames de vache
- ☐ e5 Poils et squames de chien
- ☐ e6 Epithélium de cobaye
- ☐ e7 Excrément de pigeon

ALIMENTS

- ☐ f1 Blanc d'œuf
- ☐ f2 Lait de vache
- ☐ f3 Poisson (morue)

ANTIGÈNES TUMORAUX

- ☐ CEA
- ☐ α Fœtoprotéine
- ☐ β HCG
- ☐ β 2 Microglobuline
- ☐ Ferritine

EXPLORATION THYROÏDIENNE

- ☐ Anticorps thyroïdiens T
- ☐ Anticorps thyroïdiens M
- ☐ T4 libre ☐ TSH
- ☐ T3 R.U. ☐ T3 libre

AXE HYPOPHYSO-GONADIQUE

- ☐ FSH ☐ Testostérone
- ☐ LH ☐ Testost. libre
- ☐ Œstradiol
- ☐ Progesterone
- ☐ Prolactine

GROSSESSE: DIAGNOSTIC

- ☐ β HCG (qualitatif urgent)
- ☐ β HCG (quantitatif)

GROSSESSE: SURVEILLANCE (1er T)

- ☐ β HCG ☐ Progestérone
- ☐ SP1 ☐ Œstradiol

GROSSESSE: SURVEILLANCE (2e et 3e T)

- ☐ Œstriol ☐ HPL ☐ SP1
- ☐ Progestérone ☐ α foetoprotéine (14-22 sem.)

FONCTION CORTICO-SURRENALIENNE

- ☐ Cortisol (9 h)
- ☐ Cortisol (15 h)
- ☐ DHEA (sulfate)
- ☐ Aldostérone

RENSEIGNEMENTS CLINIQUES

D.D.R.: .....

But de l'exploration - traitement:

.....

.....

.....

.....

EXAMENS URINAIRES

TESTS SEMI-QUANTITATIFS SUR ÉCHANTILLON

- ☐ Glucose
- ☐ Protéines
- ☐ Sang
- ☐ Corps cétoniques
- ☐ Bilirubine
- ☐ Urobilinogène
- ☐ Electro des protéines (sur urine concentrée)
- ☐ Test de grossesse

EXAMENS MICROSCOPIQUES ET BACTÉRIOLOGIQUES

- ☐ Examen microscopique
- ☐ Culture et numération
- ☐ Antibiogramme
- ☐ Culture BK

DOSAGES SUR URINES DE 24 HEURES

- ☐ Chlore
- ☐ Sodium
- ☐ Potassium
- ☐ Calcium
- ☐ Phosphore
- ☐ Urée
- ☐ Créatinine
- ☐ Acide urique
- ☐ 17 Céto totaux
- ☐ Cortisol
- ☐ Adrénaline
- ☐ Noradrénaline
- ☐ Dopamine
- ☐ 17 Céto fractionnés
- ☐ 17 Hydroxystéroïdes
- ☐ Métanéphrines
- ☐ VMA

EXAMENS DES SELLES

- ☐ Culture
- ☐ Antibiogramme
- ☐ Recherche de Rotavirus
- ☐ Recherche de Campylobacter
- ☐ Culture BK
- ☐ Recherche de parasites (après enrichissement)
- ☐ Sang
- ☐ Digestion

EXAMENS MICROBIOLOGIQUES

NATURE DE L'ÉCHANTILLON: .....

.....

- ☐ Examen direct
- ☐ Culture aérobie
- ☐ Culture anaérobie
- ☐ Culture mycose
- ☐ Culture BK
- ☐ Antibiogramme

Renseignements cliniques - Autres examens

Nom, prénom, nº d'identification
signature du médecin demandeur

Date.....

PROTOCOLES

Exemple de réponse envoyée aux médecins (résultats et commentaires).

LABORATOIRE DE BIOLOGIE MEDICALE

Rue des Conceptionnistes 3 - 1400 Nivelles - Tél. 067/210785

Votre référence:
Notre référence : 009110
Date de réception: 31/07/87
Date d'expédition: 07/09/87

Biologie : G. Janssens & R. Dubois
Radio-immunologie : B. Massant

Wautié Paul
Ch. de Liège 4
4150 Nandrin

Sexe : masculin
né le 05/12/61 (25 ans)

Dr. Van Hooff Jean-Baptiste
Bd Emile Bockstael 90-92
1020 Bruxelles

ANALYSES	RESULTATS	UNITES	VALEURS DE REFERENCE	HORS NORMES
HEMATOLOGIE				
Hémoglobine	16.3	G/DL	13.0 - 18.0	
Globules Rouges	5.27	Mill./MM3	4.50 - 5.90	
Hématocrite	46.9	%	40 - 54	
V.G.M.	89	Micron3	82 - 98	
CCMH	35.2	%	32.0 - 37.0	
Globules blancs	7300	1000/MM3	4000-10000	
Formule				
Neutrophiles	42	%	45 - 70	*
Eosinophiles	5	%	1 - 4	*
Basophiles	0	%	0 - 1	
Lymphocytes	45	%	20 - 40	*
Monocytes	8	%	3 - 7	*
Morphologie Erythr.	Normal			
Morphologie Leuco.	Normal			
Plaquettes	174	1000/MM3	140 - 340	
Fer	103	MicroG/DL	70 - 180	
BIOCHIMIE USUELLE				
Glucose	0.93	G/L	0.60 - 1.10	
Hémoglob. Glycosylée	7.6	%	5.0 - 8.0	
Urée	0.30	G/L	0.20 - 0.50	
Créatinine	9.4	MG/L	9.0 - 15.0	
Acide Urique	60	MG/L	30 - 70	
ENZYMOLOGIE				
Gamma GT	10	U/L	< 26	
Phosphatases Alcal.	61	U/L	60 - 170	
G O T	21	U/L	< 18	*
G P T	58	U/L	< 22	*
C P K	30	U/L	< 80	
LIPIDES				
Triglycérides	0.66	G/L	0.40 - 1.70	

LABORATOIRE DE BIOLOGIE MEDICALE

Rue des Conceptionnistes 3 – 1400 Nivelles – Tél. 067/210785

Biologie : G. Janssens & R. Dubois
Radio-immunologie : B. Massant

Votre référence :
Notre référence : 009110
Date de réception : 31/07/87
Date d'expédition : 07/09/87

Wautié Paul
Ch. de Liège 4
4150 Nandrin

Sexe : masculin
né le 05/12/61 (25 ans)

Dr. Van Hooff Jean-Baptiste
Bd Emile Bockstael 90-92
1020 Bruxelles

ANALYSES	RESULTATS	UNITES	VALEURS DE REFERENCE	HORS NORMES
Cholestérol	1.60	G/L	1.40 - 2.60	
Cholestérol - HDL	0.53	G/L	> 0.40	
Apo-lipoprotéine A1	1.57	G/L	1.00 - 2.15	
Apo-lipoprotéine B	0.79	G/L	< 1.55	
IONS				
Sodium	145	Méq/L	135 - 155	
Chlore	100	Méq/L	95 - 105	
Potassium	4.3	Méq/L	3.6 - 5.5	
Calcium	4.8	Méq/L	4.3 - 5.2	
Phosphore	37	MG/L	25 - 48	
PROFIL PROTEIQUE				
Albumine	95	%	75 - 125	
Alpha 1 Antitrypsine	80	%	70 - 125	
Orosomucoïde	78	%	65 - 125	
Haptoglobine	57	%	50 - 140	
Transferrine	83	%	75 - 125	
C'3	86	%	70 - 140	
I G G	93	%	65 - 135	
I G A	176	%	50 - 175	
I G M	51	%	50 - 160	

* * * \

LABORATOIRE DE BIOLOGIE MEDICALE

Rue des Conceptionnistes 3 - 1400 Nivelles - Tél. 067/210785

Biologie : G. Janssens & R. Dubois
Radio-immunologie : B. Massant

Votre référence:
Notre référence : 009110
Date de réception: 31/07/87
Date d'expédition: 07/09/87

Wautié Paul
Ch. de Liège 4
4150 Nandrin

Sexe : masculin
né le 05/12/61 (25 ans)

Dr. Van Hooff Jean-Baptiste
Bd Emile Bockstael 90-92
1020 Bruxelles

ANALYSES	RESULTATS	UNITES	VALEURS DE REFERENCE	HORS NORMES
SERO-IMMUNOLOGIE				
Sigma				
V. D. R. L.	Négatif		Négatif	
URINES : TESTS CHIMIQUES				
Glucose	0	G/L	0	
Hémoglobine	0			
Protéines	0	G/L		

G. Janssens & R. Dubois

Protocole complet

LABORATOIRE DE BIOLOGIE MEDICALE

Rue des Conceptionnistes 3 - 1400 Nivelles - Tél. 067/210785

Biologie : G. Janssens & R. Dubois
Radio-immunologie : B. Massant

Votre référence:
Notre référence: 009110
Date de réception: 31/07/87
Date d'expédition: 07/09/87

Wautié Paul
Ch. de Liège 4
4150 Nandrin

Sexe : masculin
né le 05/12/61 (25 ans)

Dr. Van Hooff Jean-Baptiste
Bd Emile Bockstael 90-92
1020 Bruxelles

ANALYSES	RESULTATS	UNITES	VALEURS DE REFERENCE	HORS NORMES
DOSAGES RADIO-IMMUNOLOGIQUES				
Hématologie				
Ferritine	50	NG/ML	25 - 430	
Tests thyroïdiens				
T4 libre	13.5	PG/ML	8 - 20	
T3 libre	3.3	PG/ML	1.4 - 4.1	
TSH	2.2	MicroUI/ML	0.4 - 4.0	

Exploration thyroidienne:

	T4 libre (pg/ml)	T3 libre (pg/ml)	TSH (mmU/ml)
Hypo	< 8	< 1.4	> 4.0
Hyper	> 20	> 4.1	< 0.4

Hormonologie			
Insuline (à jeun)	9.2	MicroU/ML	< 20
Allergologie			
IGE	24	UI/ML	

Chez l'adulte:
- Si < 20 UI/ml : origine atopique peu probable
- Si >100 UI/ml : origine atopique probable (si infection parasitaire exclue).

B. Massant

Protocole complet

COURRIER DIVERS

Nivelles, septembre 1986

paul wautié

chsse de liege, 4

4150 nandrin

Messieurs,

Concerne : Demande de documentation technique sur vos automates d'analyses médicales.

En raison d'une nouvelle informatisation, nous souhaiterions recevoir la documentation technique concernant les possibilités et les spécificités du ON-LINE (bi-directionnel si possible) de vos machines d'analyses médical les plus vendues.

Cette documentation devrait comporter : le type d'interface utilisé, les formats des données et des résultats, les conditions d'utilisation, etc...

Cette documentation nous est indispensable pour permettre une informatisation aussi globale que possible. En outre, ceci faliciterais les connections de vos machines, si l'achat en était fait.

Dans l'attente d'une réponse, je vous prie, messieurs, d'agreer mes salutations distinguées

P. Wautié

LABORATOIRE DE BIOLOGIE MÉDICALE APPLIQUÉE
S.P.R.L.
BIOLOGISTES : G. JANSSENS - R. DUBOIS
—
RUE DES CONCEPTIONNISTES, 3
1400 NIVELLES
—
TÉL. 067/21 07 85

Nivelles, le 4 Novembre 1986

ANALIS

Rue dewez , 14
5000 Namur

Monsieur le Directeur,

Cette missive est notre ultime espoir...

Actuellement, le laboratoire modernise son informatisation et nous sommes chargés de sa réalisation. Nous devons en outre implémenter un programme de connection bi-directionelle entre le VAX_750 et les automates d´analyses médicales prévus à cet effet : il serait regrettable de devoir modifier ce programme à l´achat de nouvelles machines d´analyses. C´est la raison pour laquelle nous avions demandé à vos responsables, Monsieur STUKKENS et Monsieur SIMON, la documentation technique concernant les interfaces et les formats des données de vos machines Coulter et Beckman. Devant la réticence manifeste à nous fournir cette documentation, nous avons envoyé une lettre signée de Monsieur JANSSENS, directeur, vous assurant de la confidentialité de la documentation qui nous serait communiquée. Malgré ceci et de nombreux échanges téléphoniques, aucune suite favorable n´a été donnée à notre requête.

Je me permets pourtant de vous rappeler que nous sommes propriétaires d´un Coulter S7 ainsi que de son interface PP Printer II et ,à ce titre, nous estimons donc que la documentation inhérente à ces deux machines nous est pour le moins acquise.

Dans l´attente de recevoir cette documentation si précieuse, je vous prie, Monsieur le Directeur, d´agréer mes salutations distinguées

G.Janssens
Directeur

P.Wautié
informaticien